FINAL REPORT

January 5, 1983

"A Study of Real-time Computer Graphic Display Technology for Aeronautical Applications"

NASA Research Grant NSG-1355

S. A. Rajala
and
L. R. Blume

(NASA-CR-169828) A STUDY OF REAL-TIME COMPUTER GRAPHIC DISPLAY TECHNOLOGY FOR AERONAUTICAL APPLICATIONS Final Report, Jan. 1977 - Sep. 1982 (North Carolina State Univ.) 165 p HC A08/MF A01 CSCL 09B G3/61

N83-18307

Unclas
08480

North Carolina State University
Department of Electrical Engineering
Raleigh, North Carolina 27650

with new pp. 8 3N18307

Rec'd Checked 3/31/83

North Carolina State University

Center for Communications and Signal Processing

AN INDUSTRY / UNIVERSITY COOPERATIVE RESEARCH CENTER

P. O. Box 5275
Raleigh, NC 27650
919/737-3015

PARTICIPATING ORGANIZATIONS:
National Science Foundation
Carolina Power and Light Co.
Digital Equipment Corp.
EXXON
General Telephone and Electronics
International Business Machines
International Telephone and Telegraph
Western Union
Westinghouse

March 11, 1983

NASA Scientific and Technical
Information Facility
P.O. Box 8757
Baltimore/Washington International Airport, MD 21240

Gentlemen:

The final report for NASA-sponsored research grant NSG 1355, entitled "A Study of Real-Time Computer Graphic Display Technology for Aeronautic Application," was submitted to your office last month. The two copies you received inadvertently contained some copyrighted software in Appendix B. Attached to this letter are two new copies of Appendix B without the copyrighted software.

Please replace the Appendix B in the original document with the new Appendix B, and discard the original Appendix B.

If you have any questions, please feel free to call me at 919 737-2336.

Sincerely,

Sarah A. Rajala

Sarah A. Rajala
Associate Professor

Attachments

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

I. INTRODUCTION

The primary goal of the research performed under this grant was the design and implementation of hardware, algorithms and software for real-time raster graphics. This work began in January of 1977 and has continued through September, 1982.

This report will summarize the research activities of the previous four years (more details can be found in the Semi-annual Progress Reports), and detail the work of the last year.

II. HARDWARE DEVELOPMENT

The philosophy of this work was to develop a raster graphic display system more powerful than was likely to be needed onboard an aircraft. The original system was intended as a research tool for the development of raster graphic algorithms, both for aeronautical and other graphic applications.

The initial hardware design included:

(1) Bipolar bit-slice microprocessor (2900 series LSI devices) for scan conversion and other processing. This gave a significant performance advantage over MOS microprocessors for several reasons: (i) a 200 ns. cycle time for the bipolar processor, vs. 500 ns. or greater for typical MOS processors; (ii) as configured this system has a 32 bit word length, not available in MOS microprocessors at the time; (iii) microprogramability allowed implementation of scan conversion algorithms with significantly fewer processor cycles than a fixed instruction set.

(2) A frame buffer constructed from state-of-the art 16K MOS dynamic RAM chips.

(3) An architecture in which all system components communicate over a fast bus: 100 ns, 32 data lines, 24 address lines, (see figure II.1).

(4) Use of microprogramming in other system components besides the bit-slice scan conversion processor.

The display controller or video sequencer, for example is independently microprogrammed, allowing selection of different numbers of pixels per line, lines per video frame, bits per pixel, and refresh from a single or "ping-pong" buffer. Once initialized, it runs independently of the main processor. The microsequencers on the frame buffer cards allow operations such as

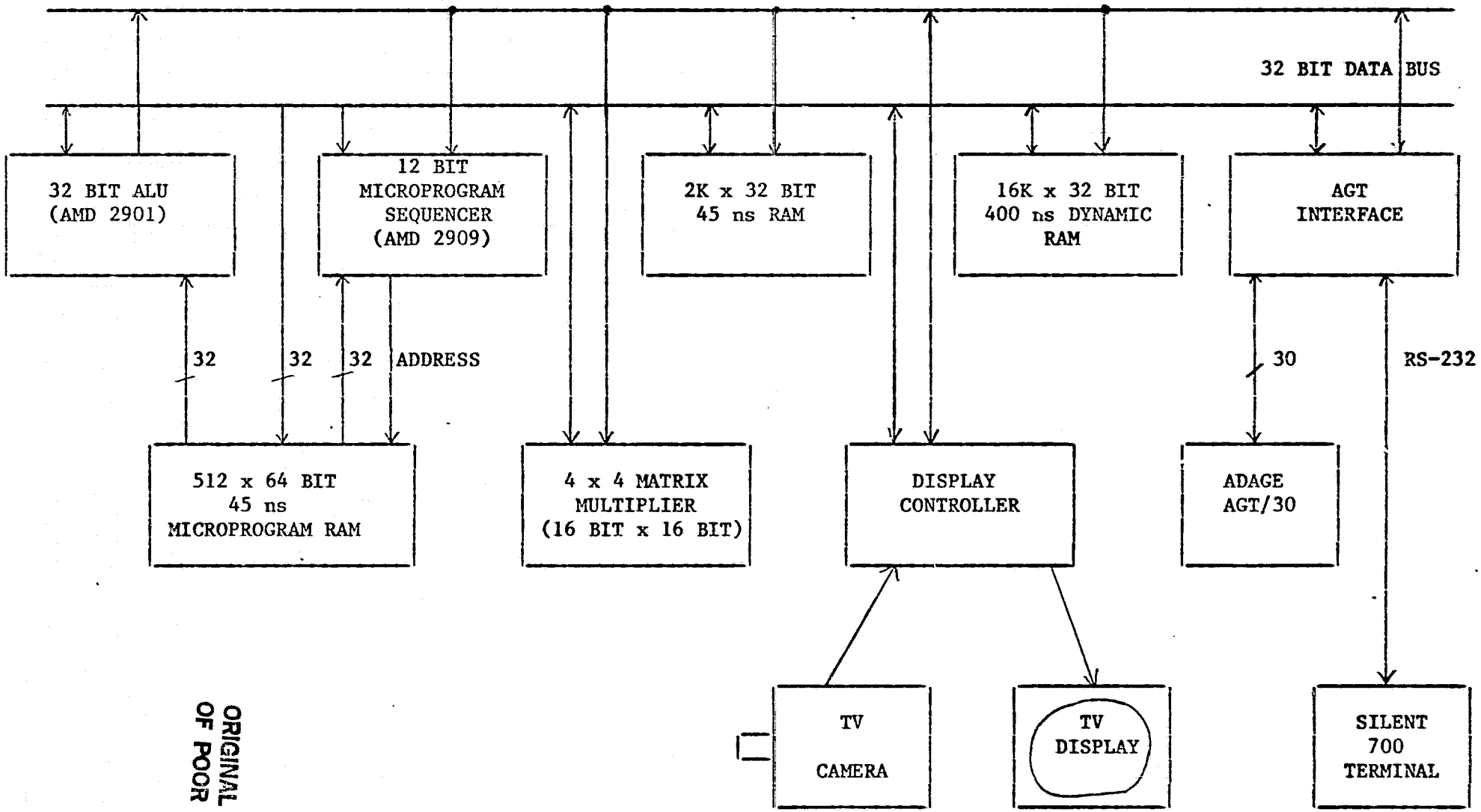


Figure II.1

Display Processor Block Diagram

read, write, zero a location, OR into a location, and read then zero a location. The matrix multiplier cards are also microprogrammed. This architecture relieves the main processor and the bus from extra cycles which would otherwise be required to control operations such as display refresh.

The initial frame buffer memory was expanded from two bits to eight bits during 1979. This expansion enhanced the capabilities of the display system in several ways:

(1) 512 x 512 eight bit display of gray scale images for sky-ground shading, anti-aliasing, etc;

(2) 1024 x 1024 two bit display of increased resolution vector displays (which again contributes to reduction of aliasing problems);

(3) 512 x 512 display incorporating ultra-fast shading of outlined areas.

Four very powerful processing elements were also added to the system. These are the multiplier-accumulator modules which each contain a 16 x 16 bit 200 nsec multiplier, a 32 bit accumulator, input and output memories, and a microprogrammed controller (see figure II.2). The controllers are relatively simple, but still allow a great deal of computational and control burden to be lifted from the 2900 system main processor.

The most significant use of the M-A modules is for coordinate transformation. A 3-D transformation can be accomplished in under 3.4 microseconds by a multiplier-accumulator module. Since all four modules contain their own controller, sub-microsecond transformation can be obtained by parallel operation. Again, the decision made early in the project to use distributed control structure with semi-intelligent processing and memory elements has proved valuable. This control has allowed the incorporation of speedy and flexible operation into the display system as a whole.

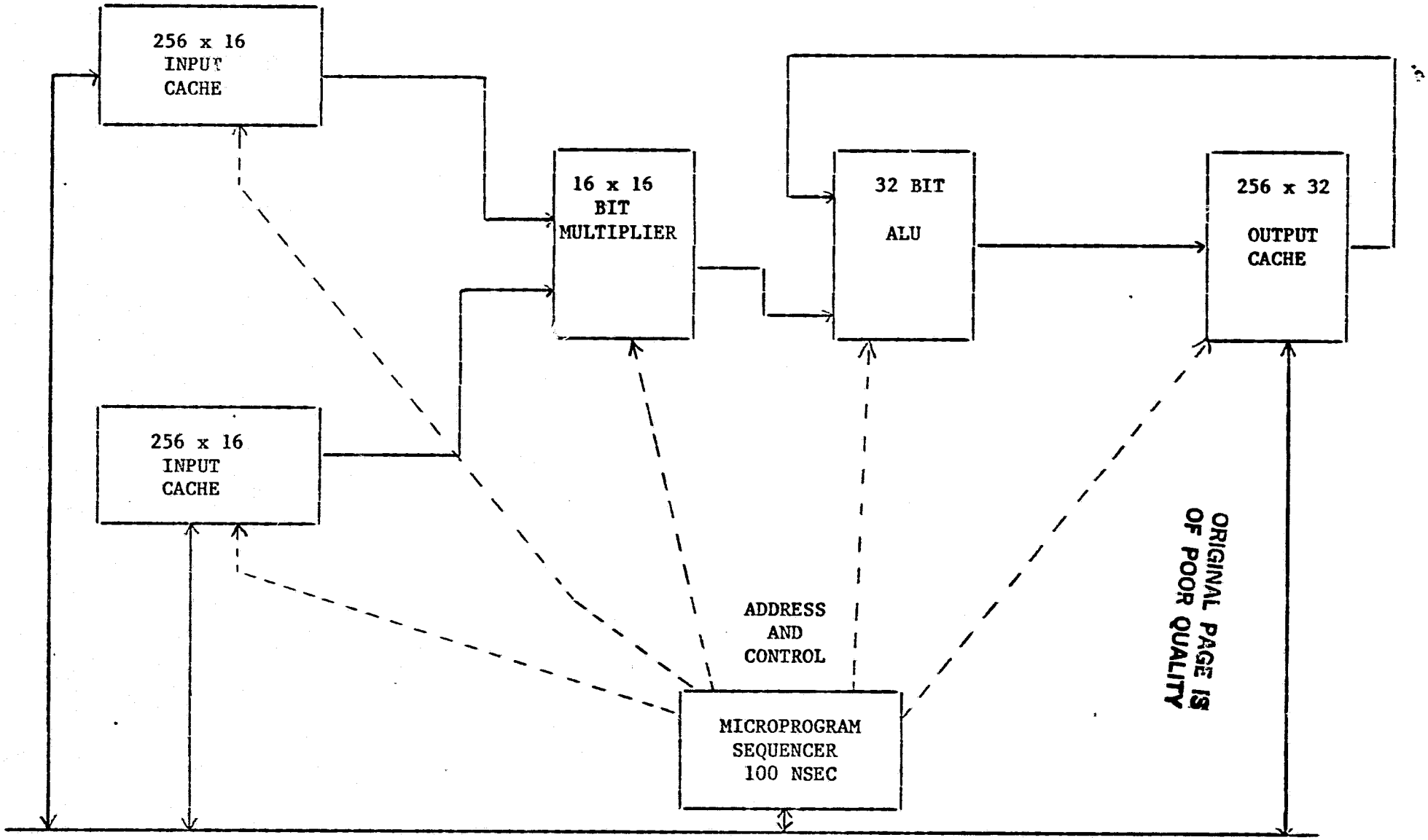


Figure II.2

Multiplier-Accumulator Block Diagram

It should be noted, that this initial prototype system led to the development of a commercial product and the founding of IKONAS Graphics. IKONAS has recently been purchased by Adage, Inc. The system has evolved several generations since the prototype and the newest system is a very powerful graphics processor system.

III. ALGORITHM AND SOFTWARE DEVELOPMENT

III. 1. Bresenham's Algorithm

A version of Bresenham's algorithm [1] for scan conversion of straight vectors was microcoded for the display processor. Figure III.1 - III-3 is a flowchart of the original algorithm. The algorithm initially waits for the video sequencer to switch to the opposite buffer of the pair used in ping-pong fashion. it then begins accessing a vector list which the host computer has loaded into scratchpad RAM. A word consisting of all ones signals the end of the vector list. Here, however, x and y are 9 bits each for a 512 x 512 display. When adjoined, they form and 18 bit pixel address. The algorithm identifies a pair of coordinates specifying a vector. It then determines which octant the vector is in, and sets up the appropriate horizontal or vertical and diagonal displacements. The actual iteration takes only four processor cycles per pixel assuming that the frame memory is not busy when the write is attempted.

III.2 Cross-assembler and Loader

A cross-assembler was originally written on the Adage AGT/30 host computer to ease the task of writing microcode for the display processor. The cross-assembler was written in FORTRAN with as little use of non-ANSI standard constructs as feasible, so that transfer to another host computer involves only minor changes. Since the original cross-assembler was written several updated versions have been written by IKONAS.

The output of the cross-assembler consists of an object code disk file, which is later read by the loader. A listing showing the source and object code side by side is optional. A companion loader program down-loads object files generated by the cross-assembler into the display processor's writeable control store.

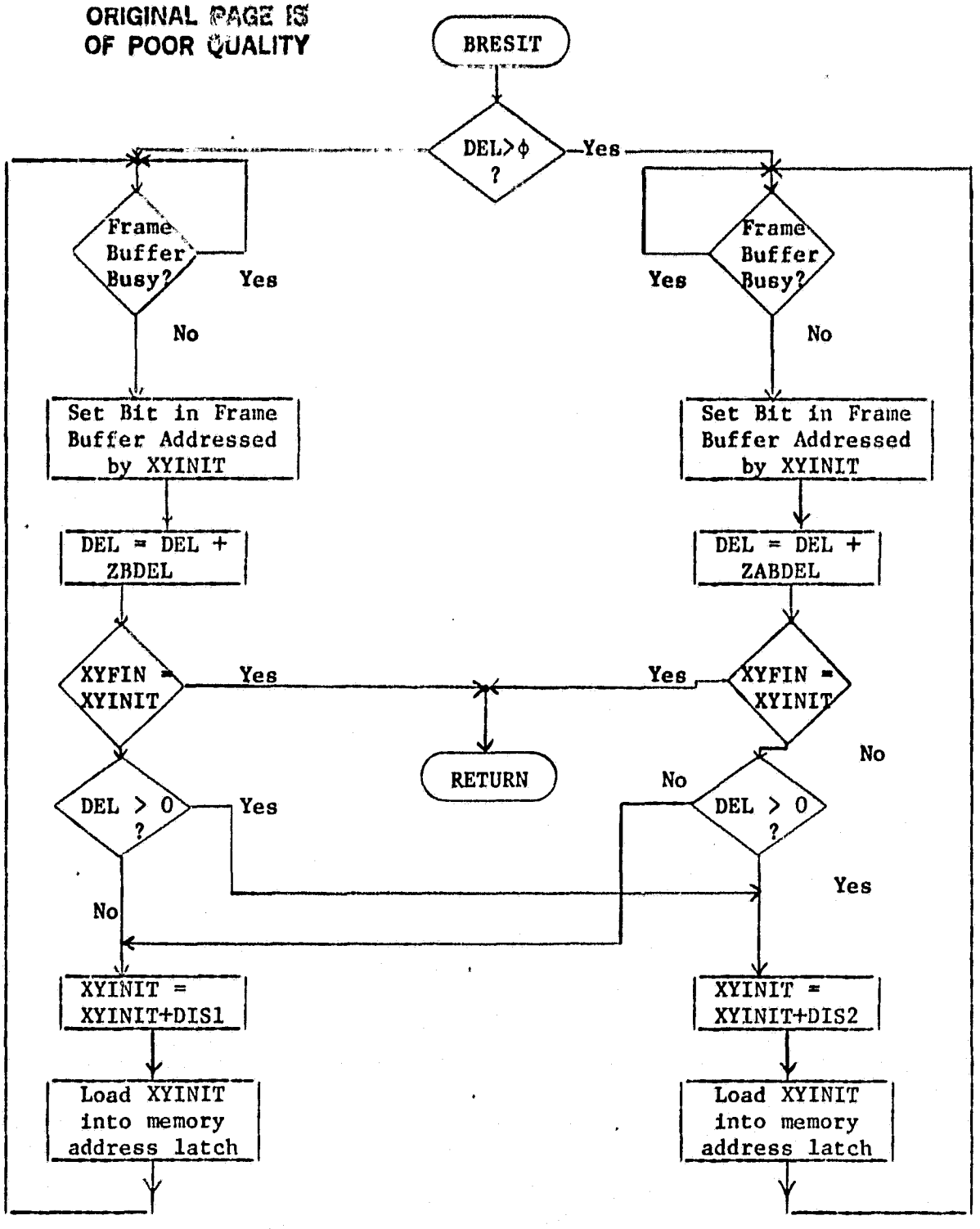


Fig. III.1 Bresenham Algorithm Flowchart

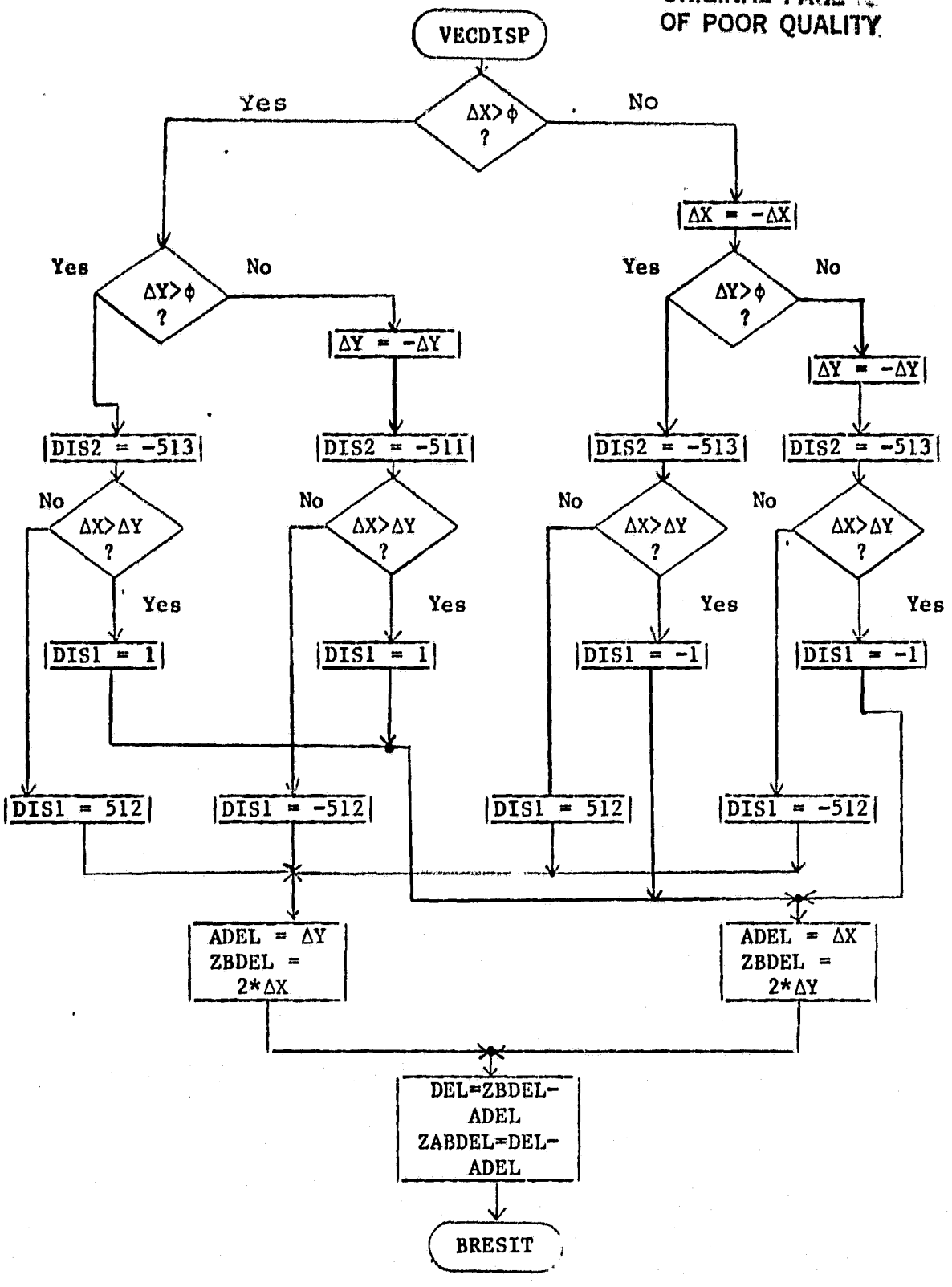


Figure III.2
Bresenham Algorithm Flowchart

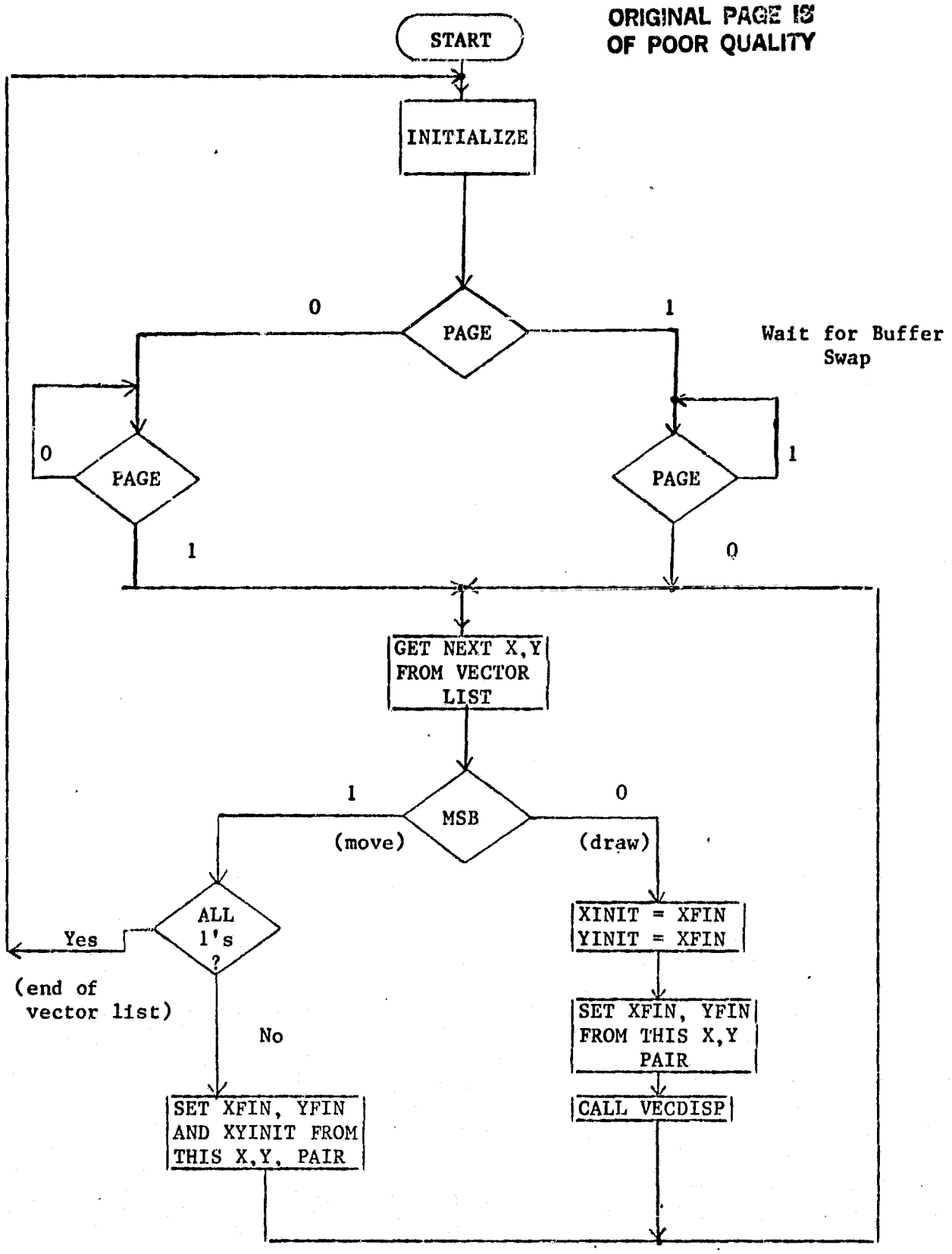


Figure III.3
Bresenham Algorithm Flowchart

IV. APPLICATIONS SOFTWARE

IV.1 Electronic Attitude Direction Indicator

An Electronic Attitude Direction Indicator (EADI) demonstration program for an AGT was obtained from NASA-LRC for use in demonstrating the raster graphic display system. In this program, an EADI display is driven by simulated DC-8 dynamics. The user "flew" using the AGT's joystick. For this demonstration, the following modifications were made to the EADI demonstration program: The sky-ground shading was removed. This was done for several reasons. Only a single intensity level was available at the time of the demonstration and the sky-ground shading would obscure the other display features. In addition, all the necessary vectors would probably not have been scan-converted in one frame time. Besides, there are more efficient ways to generate shaded areas, as described later. All character items were also removed.

IV.2 Windowing

Windowing in this context refers to the process of displaying a sub-area of a considerably larger image. This is a necessary part of moving-map type cockpit displays.

A window size and origin are chosen. With reference to figure IV.1, each vector is tested to determine whether it is entirely within one of the regions R1, R2, R3, or R4. This is done by comparing vector end points within window boundaries. If the window size is small with respect to the size of the total image and assuming a random distribution of vectors, on the average 75% or more of the vectors are eliminated. One then scan converts the remaining vectors and windown each pixel.

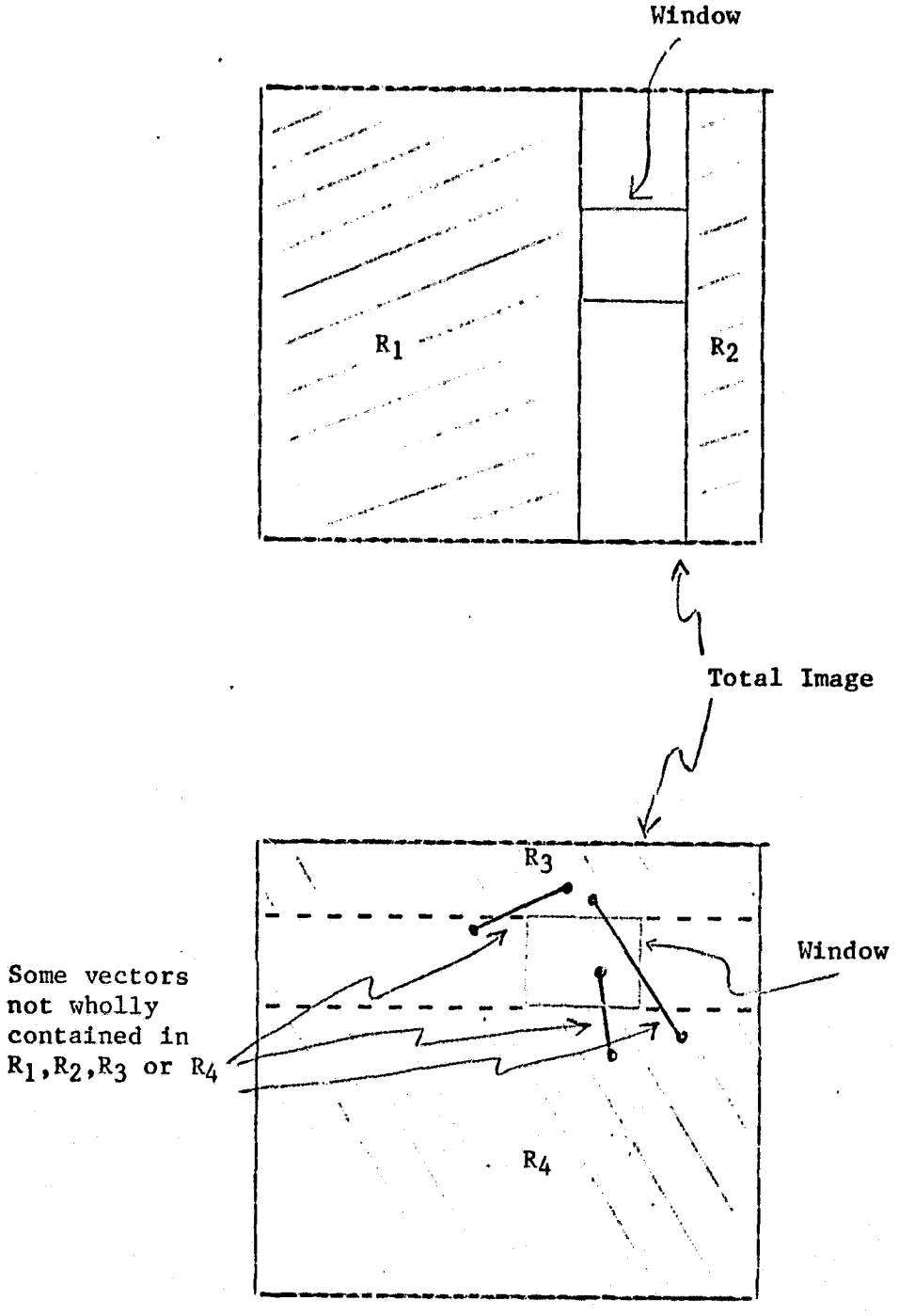


Figure IV.1
Illustration of a Windowing Algorithm

IV.3 High-Speed Fill

The generation of shaded areas is awkward on vector displays, requiring a large number of closely-space vectors. A specific application is the sky-ground shading in an EADI display. The area to be shaded is very large and its boundaries change dynamically. In addition, the intensity level of the sky and ground must be lower than that of the line images so as not to obscure them.

An algorithm was proposed for doing sky-ground shading. The important concept in this algorithm is not the fact that it fills, but that a pixel is divided into a 'priority' field and a 'fill' field. Information from the priority field takes precedence over the data supplied from the fill circuit, which is controlled by the fill field.

The reason the priority field approach is important to this research is that many of the proposed figures in the display are outlined in one color and filled with another. Using a priority field makes it possible to describe an outlined, filled figure with a single boundary line, where the outline data is in the priority field and the fill data is in the fill field.

It should be also noted that lines which only have data in the priority field may be drawn through filled figures without affecting the filled process, yet they will be visible on the display. Figure IV.2 shows a general priority field circuit. It permits the priority field to be defined by means of a mask which can be loaded by software.

The approach chosen in the design of the new video control card is as follows: in one mode of operation, the signal corresponding to its shaded areas is gated by the output of a flip-flop which will be added to the video generator. The scanned, bit serial output of an extra bit plane is used to toggle this flip-flop. Thus, what is written in to this bit plane are the

edges of the shaded area. This approach is fast, because fewer write accesses to the frame buffer are required than would be if all pixels in the shaded area had to be written. A second advantage is that line generation may be used with very minor modification for shaded area edge generation. Thus, this approach is compatible with the real-time, scan-conversion per frame, ping-pong buffer mode of operation.

IV.4 Real-Time Antialiasing

Early system demonstrations revealed the well-known adverse affects of spatial sampling. One approach to the solution of this problem is simply higher display resolution and rate. There is, however, a limit on any imaging system.

Spatial filtering can diminish the aliasing effects associated with limited resolution. The multiplier-accumulator modules in the system will allow the implementation of such a filter. The filter output will have multiple intensity levels which can be handled with the extended memory.

Another approach to solving the problem of aliasing was proposed by Barros and Fuchs [2]. They presented an algorithm for generating precise, smooth images of line drawings on multi-gray-level pixel-mapped video systems. The method is based on an analysis of the boundary conditions at each pixel affected by one or more lines.

Their approach is not intended for real-time graphics, as it requires a great many computations to determine the boundary conditions. However, it has inspired a possible solution suited to real-time graphics.

If a small region of the screen is examined closely, the reason for the jaggy lines becomes apparent. The "ideal" edge of a line may often go through the area of a pixel but not cover it completely, see Figure IV.2.

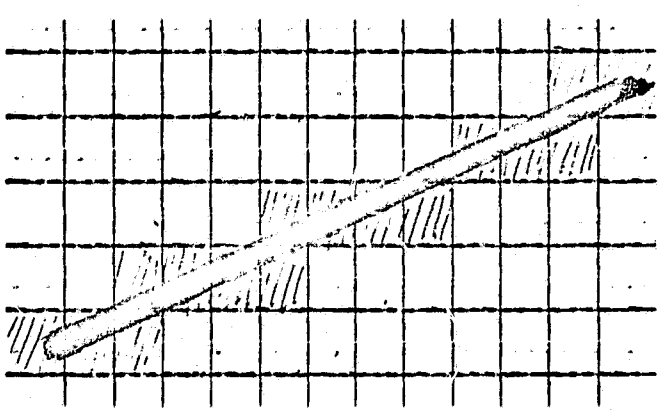


Figure IV.2

The second effect of the algorithm is found in diagonal type lines. In these lines, the adjacent pixels of the occasional M1 move are reduced in intensity. This removes the bright spots caused by mutual reinforcement between these pixels (see Figure V.5).

V.1.2.1 Analysis

Bresenham's algorithm generates a sequence of M1 and M2 moves using two routines, or the program loops, one for each move. Of these two loops only the M1 loop has additional instructions inserted in it to implement the new algorithm. This means that M2 moves takes place without loss of speed.

The basic algorithm is based on the behavior of the decision variable ∇, del, in the M1 routine. Upon entry into the M1 loop, the decision variable is some negative value. Each pass through the M1 loop, the decision variable is increased by $2\Delta b$ (see equation 2(b) in section V.1.1). Therefore, the number of M1 moves prior to the next M2 move is given by the following formula:

$$N_{M1} = \frac{\nabla}{-2\Delta b} = \text{number of M1 moves} \qquad (1.2\text{-}1)$$

A transition region can be constructed prior to the next M2 move using multiples of $-2\Delta b$. A transition region of n pixels must begin with the first pixel where del (the decision variable) is greater than or equal to $-n2\Delta b$. So long as the length of the transition is some power of two, then the test value that indicates entry into the transition region can be constructed using shifts instead of a multiply. Since the transition region is used to remove line discontinuities, its length can be constant for a wide range of line slopes.

For example, say that drawing a particular line generates line segments (on each scan line) of 20 pixels. Further, say that a transition region of 4

pixels is used. Then, 15 of the 19 times through the M1 routine, the only additional instruction executed (beyond those that would be executed for Bresenham's algorithm) is a test to see if the transition region has been entered, which fails. The fact that most of the pixels are output with little additional computation is what accounts for the speed of the algorithm.

To understand the transition region, it is easier to first consider the raster display of a line input by analog means (cameras, etc.) which forms an acute angle with the scanlines. If the line is the width of a single scanline, the intensity of the line displayed fades slowly on one scanline as it increases on the scanline above or below. The intensity transition takes place in such a way that the net intensity on the two scanlines at any point is equivalent to the maximum intensity reached on a single scanline.

In a similar fashion, the transition region of this algorithm is designed to yield a two-step approximation to the analog intensity transition. In the first step, upon entry into the transition region, the intensity of the pixels output to the current scanline is decreased by about one-third while the intensity of the pixels output to the next scanline (indicated by the M2 move) is increased accordingly to about one-third of full intensity. In the second step (halfway through the transition region), the two intensities are simply swapped so that the pixel intensity on the new (next) scan line is further increased (to about 2/3 of full intensity) while the pixel intensity on the current scanline is correspondingly decreased to 1/3 intensity. When the M2 move is made at the end of the transition region, the first full intensity pixel is output to the new scanline and no more pixels are output to the previous scanline. Selection of the intensity levels used for pixels in the transition region will be covered in the section on implementation.

Since the transition region contains two steps, it should (for symmetry) be constructed of two equal length segments. In the same way that a test constant was developed to locate the start of the transition region, one is created to locate the region's midpoint where the intensities are swapped. The formula for the starting location constant of an n pixel transition region is n times $-2\Delta b$. Because the n pixel region (by construction) contains two equal segments, there must be some integer m such that $m = n/2$. The midpoint test value is, therefore, $-m2\Delta b$ or $-n\Delta b$. It is important to remember that n is some multiple of two (preferably a power of two) because $-n\Delta b$ must be a multiple of $-2\Delta b$.

The selection of intensities and the computation of the two transition region constants take place prior to the actual line generation. Line generation takes place using the following scheme. If the M2 routine is entered (del>0), then an M2 move is made, a pixel of full intensity is output and del is updated (incremented by the constant $2\Delta b-2\Delta a$ as per equation. 1.1-2(b)).

If the M1 routine is entered, del (which must be negative at this point) is compared with the transition region start constant. If del is less than the test constant (more negative), the normal M1 sequence of operations take place -- an M1 move is made, a full intensity pixel is output, and del is updated (incremented by $2\Delta b$ as per equation 1.1-2(a)).

If del is greater than or equal to the start point test constant, but less than the midpoint test constant, then a pixel of minimum intensity is output in the M2 direction, the M1 move is made, a pixel of intermediate intensity is output, and del is updated. The only remaining possibility is that del is greater than or equal to the midpoint constant (and less than zero); in this case the preceding set of operations is carried out, but with the two intensities swapped.

All of the aforementioned procedures, with one exception, apply to both axial and diagonal type lines. When processing diagonal type lines (those with a non-negative initial del value), the only change in the previously outlined procedure is to set the minimum intensity constant to the same value as the intermediate intensity constant. Full and intermediate level pixels will be the only ones used in these lines. The explanation for this is as follows.

It can be shown that diagonal type lines contain, at most, singular M1 moves between sequences of M2 moves and that (assuming the transition region has been set to a length greater than or equal to the minimum of two pixels) del will always fall within the second step of the transition region when the M1 loop is entered. Consequently, for diagonal type lines, the M1 loop would:

1) output a 2/3 intensity pixel in the M2 direction;

2) output a 1/3 intensity pixel in the M1 direction, and

3) return to the M2 loop.

The result would be an incorrect line, of the type pictured in Figure V.6.

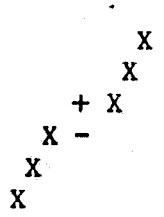


Fig. V.6 Incorrectly drawn diagonal type line caused by using minimum intensity pixels.

Instead for diagonal lines, by setting the minimum intensity constant (output in step 2 above) equal to the intermediate intensity constant, a line with a proper transition region is output (see Figure V.5(b)).

The algorithm will thus produce a series of M2 line segments, which lap by two pixels and begin and end with intermediate intensity pixels (except for the end points of the first and last segments). It will not produce any full intensity pixels which are M1 adjacent. This prevents mutual reinforcement and gives the appearance of uniform line intensity. The lack of minimum intensity pixels is not detected by the eye due to the tendency of the entire diagonal line to appear less intense.

Symmetry was bypassed earlier, but it should be obvious that if a line is constructed in the first octant, and the transition regions precede the M2 transitions, that these regions will be on the left side of the M2 transitions. Since constructing the same line in the fifth octant by reversing the end points will result in transition regions on the right of the M2 transitions, these lines would not look the same. A method of relocating the transition region centers so that they occur near the previous position of the adjacent M2 transitions is needed to give lines drawn in opposite directions a similar appearance.

The problem is solved by simply shifting the M2 transitions of axial type lines in the direction of the M1 transition by half the number of pixels in the transition region. It should be noted that diagonal type lines do not require symmetric correction because of the degenerate nature of their transition regions.

Refering back to the formula (1.2-1) used to compute the number of M1 moves until an M2 move, and letting m equal half the number of pixels in the transition region, it should be apparent that adding $-m2\Delta b$ to the initial del

(which is negative for axial lines -- the type needing symmetric correction) will produce m additional M1 transitions at the beginning of the line. It can also be shown that after these m additional transitions, del will be updated by $m2\Delta b$ which will return it to its initial value. This will insure that the M2 transitions will occur normally, but shifted toward the line's ending point.

Because the total number of moves (both M1 and M2) used to construct a line must be Δa (as specified by Bresenham's algorithm), the addition of m M1 moves to the beginning of a line results in the loss of m moves from the end of the normal sequence. While this truncation usually produces the desired line, improper application of this symmetric correction can result in the distortion of the line being displayed and failure to arrive at the specified ending point. While the length of the transition region is constant for a wide range of line slopes, if an excessively long transition region is chosen arbitrarily for a line which consists of very short M1 move runs between the M2 moves, then the line will appear to be a straight line segment (in the M1 direction) followed by one having the correct slope but incorrect end point. Hence, a method is needed to insure that the transition region length is appropriate.

In his paper, Bresenham has derived the following relationships. All lines produced by his algorithm are made of runs of the dominant move (M1 for axial lines) which, except for the first and last run, have a length of either Q or Q-1 (Q is an integer whose value is related to the slope of the true line). He has also shown that the first and last runs have length M or M-1 where $M = Q/2$.

Using these relationships, it is possible to use the initial value of del (before correction) to determine whether the transition region length selected

is appropriate. If equation 1.2-1 was applied to this initial del, the number of M1 moves prior to the next M2 move would be approximately Q/2. If a transition region length of 8 was used for a line where Q equals two, every time the M1 routine was entered only the second half of the transition region would be produced. This is because the M1 run, of length Q=2 or Q-1=1, would, being less (in magnitude) than the midpoint test value of 4, fall entirely in the second half of the transistion region. The correction for symmetry would distort the line as well. By comparing the symmetric correction value, $-m2\Delta b$, with delta before correction, essentially half the length of the transition region is being compared with Q/2. Of course this is an approximation. In practice, since the transition region length is kept relatively short in order to obtain the best execution speed, it is just a matter of deciding to reduce the transition region length to a minimum value whenever ∇_j is less than half the length of the transition region. Therefore, if $-m2\Delta b$, the length of half the transition region, is less than the initial value of del, the length of the transition region is set to two.

Checking the transition region length by this method and adjusting it if necessary, assures that the ending point will be output by this algorithm. It should be obvious that if only M1 moves are lost in the sequence truncated by symmetric correction, the ending point will be reached. If only one M2 move is lost, the ending point will be output as a pixel in the transition region. When an M2 move is lost and the last move is an M1, the ending point is assured to be in the transition region because the lost M2 move must have been within m moves of the final location, because of symmetric correction. Since the transition region has a length of 2m, the transition region test in the M1 routine will be true, and a pixel (the ending point) will be output in the M2 direction. Because check and adjustment will force m to be less than or equal

to M, no more than one M2 move could be lost. This would be the case if m = M and the last run was of length M-1.

Loss of an M2 move will prevent the position datum, used by the move loops tines (M1 and M2), from indicating the ending point. For this reason, a comparison of this datum with the ending point's coordinates cannot be used to flag the end-of-line condition. The value of Δa indicates the total number of moves in a line, and is used as an end-of-line counter.

One source of asymmetric behavior could not be eliminated easily. This occurs when an axial type line contains M1 runs which have only one move. Lines in the first octant will produce an intermediate intensity over minimum

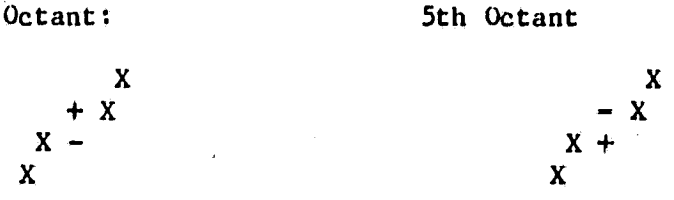


Fig. V.7 Assymetric behavior.

intensity pixel pair between two M2 moves. Whereas, the same line drawn in the fifth octant will have the intensities in such pairs reversed (see Figure V.7). Fortunately, when these type of lines are displayed on a high resolution display, one where the area of individual pixels is small, the difference is virtually imperceptable and does not degrade the lines' appearance. For this reason, attempting to correct this condition would not be productive; it is not worth the additional programming it would require.

In depth treatment of asymmetric behavior and efforts to produce a completely reversable algorithm with fully symmetric transition regions might be interesting, but this algorithm is intended as a fast, visual (not mathematically elegant) solution to aliasing. The widely used line plotting algorithm

which Bresenham published in 1965, is not reversible. Of course, the information used to derive this new antialiasing algorithm can be used to create variations on this approach. However, changes which increase the execution time, without giving significant visual improvement in the lines displayed, should be avoided.

This completes the analysis of the mechanics of the basic antialiasing algorithm. The next part of this section will cover the practical aspects of its implementation.

V.1.2.2 Implementation

Prior to executing the algorithms, two video intensity levels must be selected for each color to be used in drawing lines in the display. Visually these levels must produce two equal steps between the background level and full intensity. These levels can be derived either visually by trial and error, or from the gamma correction data for a calibrated monitor. The monitor's brightness and contrast controls must be preset to a known value because the settings will affect the numerical (D-to-A) values which produce the desired levels. Later it will be shown that these intensity levels will become the data in the system's color lookup table. The minimum and intermediate, intensity levels, are displayed for the pixel codes which are denoted 33% and 66% respectively, in the FORTRAN subroutine comments.

If the system has two different memories -- a frame buffer for storing the pixel codes and a lookup table for storing the intensity values, the pixel codes should be selected based on the numerical characteristics of their insertion into their memory. In a color display system where the lines are output in the reverse order of their proximity to the viewer (closest last), a replacement of the frame buffer data by the data output (by the algorithm) for later lines would be implemented, and random pixel codes could be used.

For another system where line intersections are to take on the maximum intensity of the crossing lines, the boolean OR operator, which is readily available on the ALU or memory system, might be used to select the maximum intensity of the intersecting lines.

The pixel codes should be selected to take advantage of this intensity OR'ing. Note that only two bits are required to represent the 3 intensity levels used by this algorithm. A slight problem arises, however, when a 2 bit representation of the intensities is used. The OR of a minimum intensity value (say 01) with an intermediate intensity value (say 10) yields a value (here 11) other than the desired intermediate intensity value. Testing has shown that in many cases using only two bits to express the pixel's intensity (the other six bits may be used to indicate color) is sufficient because the probability of ORing a minimum intensity pixel code with one for intermediate intensity is relatively small especially with hidden line removal. The visual impact of such occurrences is also small, making the prevention of this by the use of additional bits or additional programming less than productive for most applications.

The triple of pixel codes (full, plus the other two) selected for a line to be drawn would of course be passed to the routine implementing this algorithm (the INTENS array in the FORTRAN example is used for this).

The remainder of this section will concentrate on the modification of Bresenham's algorithm to implement antialiasing. Comparison of the differences between the standard Bresenham subroutine DRAW0 and the antialiasing subroutine DRAW1, both in Appendix A, will serve to emphasize the changes.

Three program variables, FULL, IMED and IMIN, set prior to line generation, represent the full, intermediate (66%) and minimum (33%) intensities,

respectively. It should be noted that IMIN will, in one case (covered later), be set to the intermediate (66%) value; this is the only exception.

There are three constants used in the FORTRAN subroutine DRAW1; these should be powers of two since they are used in multiples and should be implemented as shifts. The first constant is the number of pixels in the overlaps (transition regions) of the axial line segments (stairsteps) generated by the Bresenham algorithm. This lap constant (LAPCON (1)) is used for most of the lines which are constructed primarily of axial moves.

The second constant, also a lap constant (LAPCON (2)), is used to set a long overlap used on axial lines which form very shallow angles with the horizontal axis (or other M1 axis). These lines have long runs of M1 moves and the longer overlap enhances their appearance, giving a better visual approximation to the true line.

RATIO, the third constant, designates the aspect ratio defined as $\frac{\Delta a}{\Delta b}$ RATIO determines how great the difference between Δa and Δb must be to use the second lap constant. The following values are normally used for these constants:

Lap constant one	=	4 (pixels)
Lap constant two	=	16 (pixels)
Aspect ratio	=	32

The constants just described may be assigned other values as desired, but the order of their magnitudes must remain the same, that is, RATIO must be the largest and lap constant one must be the smallest and at least equal to two (as described in the analysis).

The necessity for making the second lap constant larger than the first is evident; however, the requirement that the aspect ratio be larger than the second lap constant may not be clear. The second lap constant is selected

when Δa is greater than or equal to the aspect ratio (RATIO) times Δb, i.e.

$$\Delta a > \text{RATIO} * \Delta b \rightarrow \text{use LAPCON2}$$

$$\Delta a < \text{RATIO} * \Delta b \rightarrow \text{use LAPCON1} .$$

It is desirable that the longer transition region length be selected only where its length will be correct, i.e. where it is less than Q. The following equations are found in Bresenham's line compaction paper [3]:

$$\nabla b = \text{MINIMUM} (\Delta b, \Delta a - \Delta b) \tag{3.2}$$

$$Q = \frac{\Delta a}{\nabla b} \tag{3.3}$$

Using equations 1.2 and 1.3, it can be shown that if Δa is greater than or equal to $2\Delta b$ then the following equation is true:

$$Q = \frac{\Delta a}{\Delta b} \tag{3.4}$$

This is the same as saying, for i which is the maximum integer where Δa equals $i\Delta b + r$, Q equals i. Therefore, if Δa is greater than or equal to RATIO times Δb, then RATIO must be less than or equal to Q. Because the longer lap constant is less than the aspect ratio, it can never produce a transition region longer than the M1 runs of a line for which it is selected.

Execution of the algorithm begins with the normal computations used in Bresenham's algorithm. The octant is established, the M1 and M2 moves are set, Δa and Δb are set, and the initial value of the decision variable, ∇_1 is computed.

The next computations are to set two test variables used to position the overlaps (transition regions), set the actual intensities to be used, and in some cases change the value of ∇_1. These steps, along with the rest of the algorithm, are pictured in the flowchart (Figure V.8). Frequent reference to the flowchart will aid in the understanding of the following discussion.

Two test variables, denoted ANTI2 and ANTI1, are used to locate the transition regions' starting and midpoints, respectively. Because the transition region is divided into two equal parts, the start point test value (ANTI2) is always set equal to twice the midpoint value (a shift left of one). The value of ANTI1, however, depends on the type of line being drawn. The setting of ANTI1 and the setting of the intensities is discussed next.

After the full (FULL) and intermediate (IMED) values are set using their respective parameters, the initial decision variable ∇_1 is checked for a non-negative value. If ∇_1 is greater than or equal to zero, then the line is of a diagonal type. For these type of lines, ANTI1 is set to its minimum value, $-2\Delta b$, and the minimum intensity variable IMIN is set to the value of the intermediate parameter (66%) (see analysis for reasoning here). Processing then proceeds to the setting of ANTI2 and the generation of pixels.

For axial type lines, a series of three cases are checked to set ANTI1. First Δa is checked to see if it is greater than or equal to the aspect ratio (RATIO) times Δb. If it is, ANTI1 is set to minus the long lap constant (LAPCON (2)) times Δb. If this first test fails, ANTI1 is set to minus the first lap constant (LAPCON (1)) time Δb. This value of ANTI1 is now compared with the initial del computed by the Bresenham algorithm. If ANTI1 is less than ∇_1, ANTI1 is set to its minimum value, $-2\Delta b$. Now that ANTI1 is set, the minimum (IMIN) intensity variable is set to 33% intensity. Next, ANTI1 is added to ∇_4 to shift the laps in axial type lines for the symmetry correction described in Sec. V.1.2. After ANTI2 is set by multiplying ANTI1 times 2, the generation of pixels can begin.

If the starting point pixel is to be output, it is now output at full intensity. Some implementations may assume the starting point was output as the ending point of the previously processed line.

A count is now initialized using the Δa value and is decremented each time M1 or M2 loop is executed. The variable containing Δa, if still available, may be used for this; it is no longer needed for other computations.

As indicated previously, this value (Δa) is the number of moves necessary to generate the line. When the count reaches zero, line generation is complete. Using Δa for an EOL counter prevents potential difficulties with using the position datum as an EOL indicator (see Sec. V.1.2).

Line generation begins now with the same loop (M1 or M2) that it would for Bresenham's algorithm and will proceed until the count (mentioned above) reaches zero. The M2 loop, exactly the same as it is for the Bresenham algorithm, outputs pixels at full intensity. The M1 loop contains the additional code for anti-aliasing.

When the M1 loop is entered, del is compared with ANTI2. If del is less than ANTI2, the M1 loop performs exactly as specified by Bresenham and outputs a new pixel at full intensity. Since del is usually more negative than ANTI2, the compare instruction is generally the only new instruction executed in the M1 loop.

If del is greater than or equal to ANTI2, then the algorithm has entered the transition region. Del is then compared with ANTI1. If del is less than ANTI1, then the midpoint of the transition region has not yet been reached, and the following sequence takes place. First, a pixel of minimum intensity is output using the current (not updated) position plus the M2 move as its location. Then, the position is updated with the M1 move and an intermediate intensity pixel is output, (see Figure V.9).

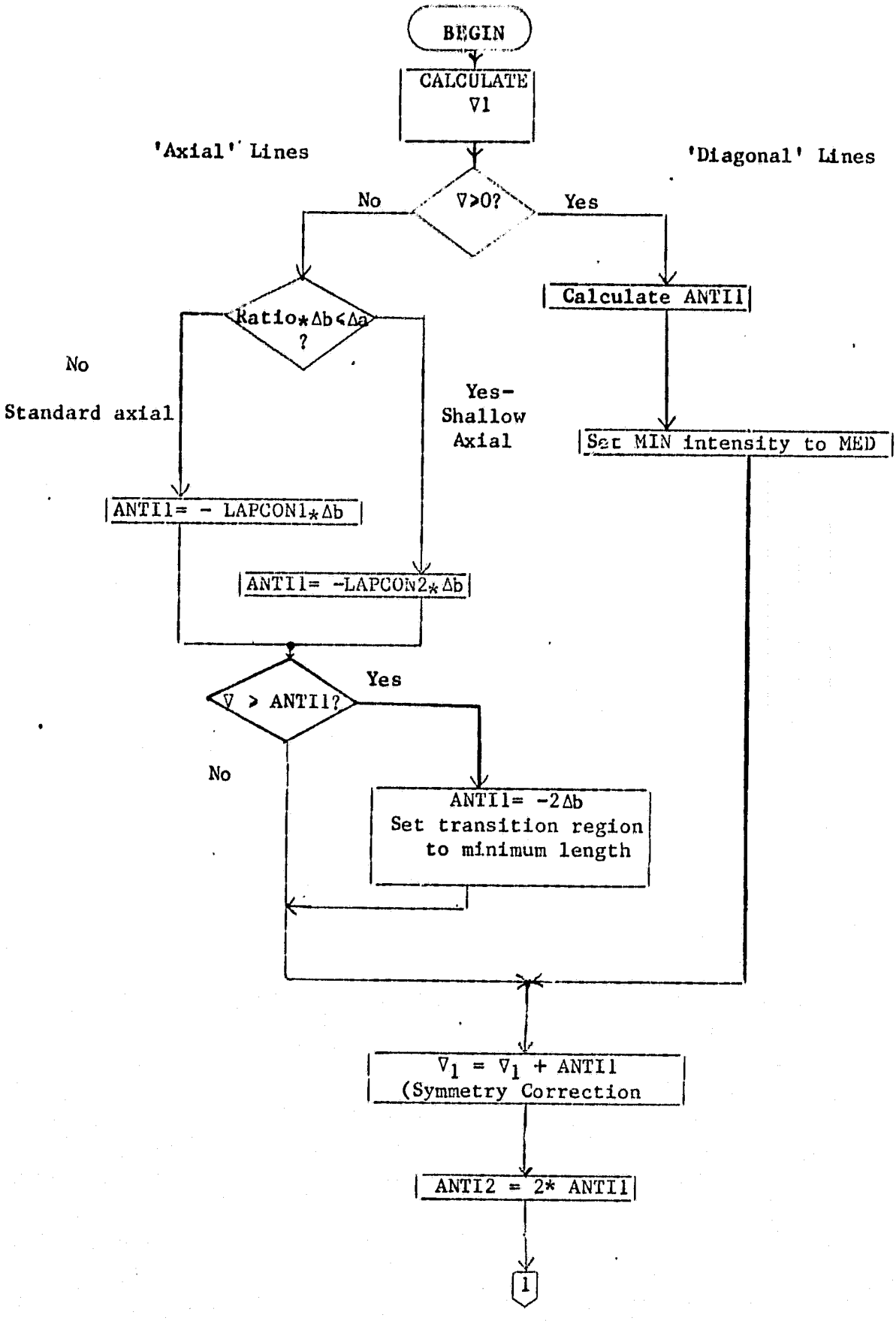


Fig. V.8 Flowchart for antialiasing algorithm.

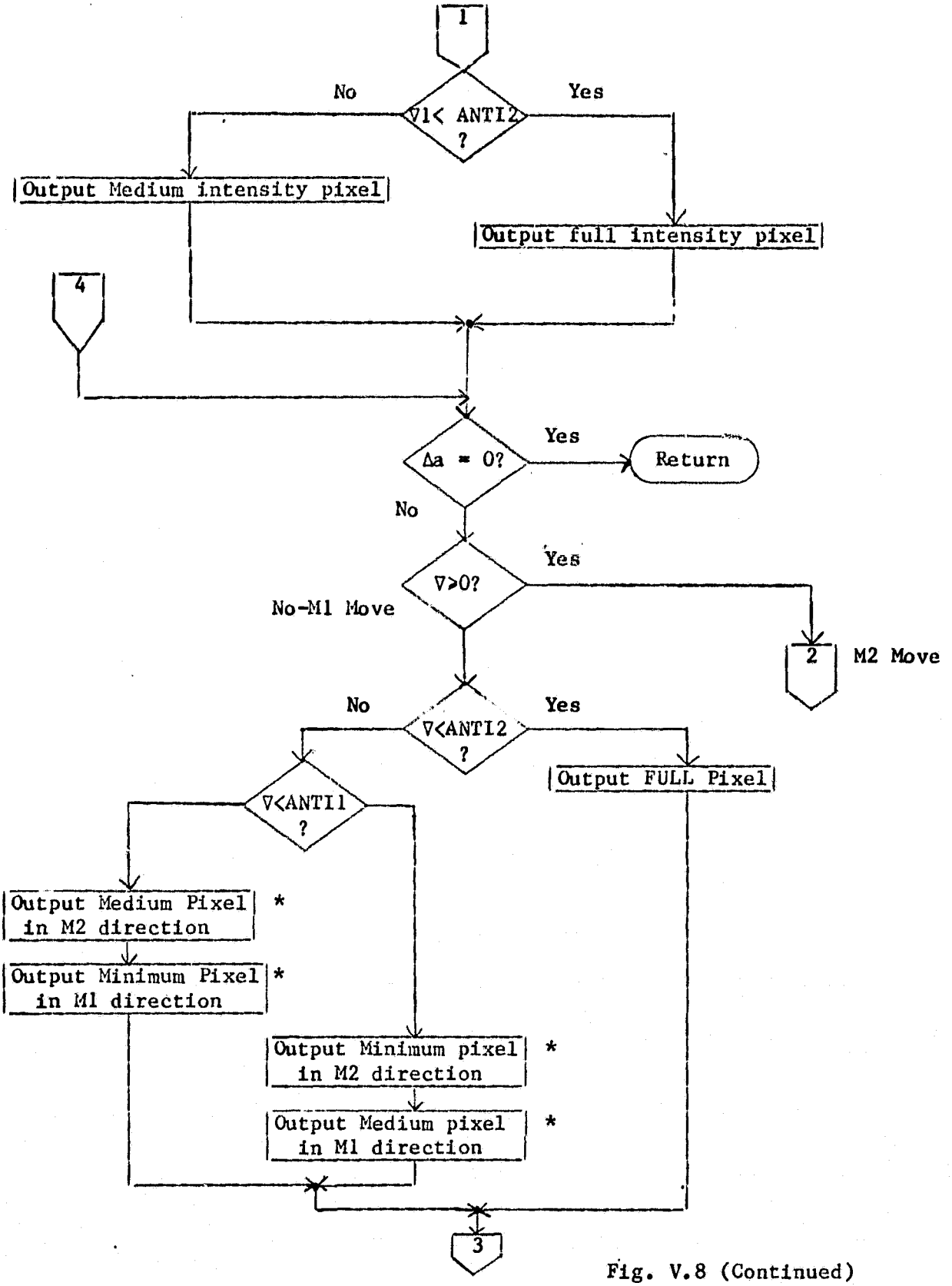


Fig. V.8 (Continued)

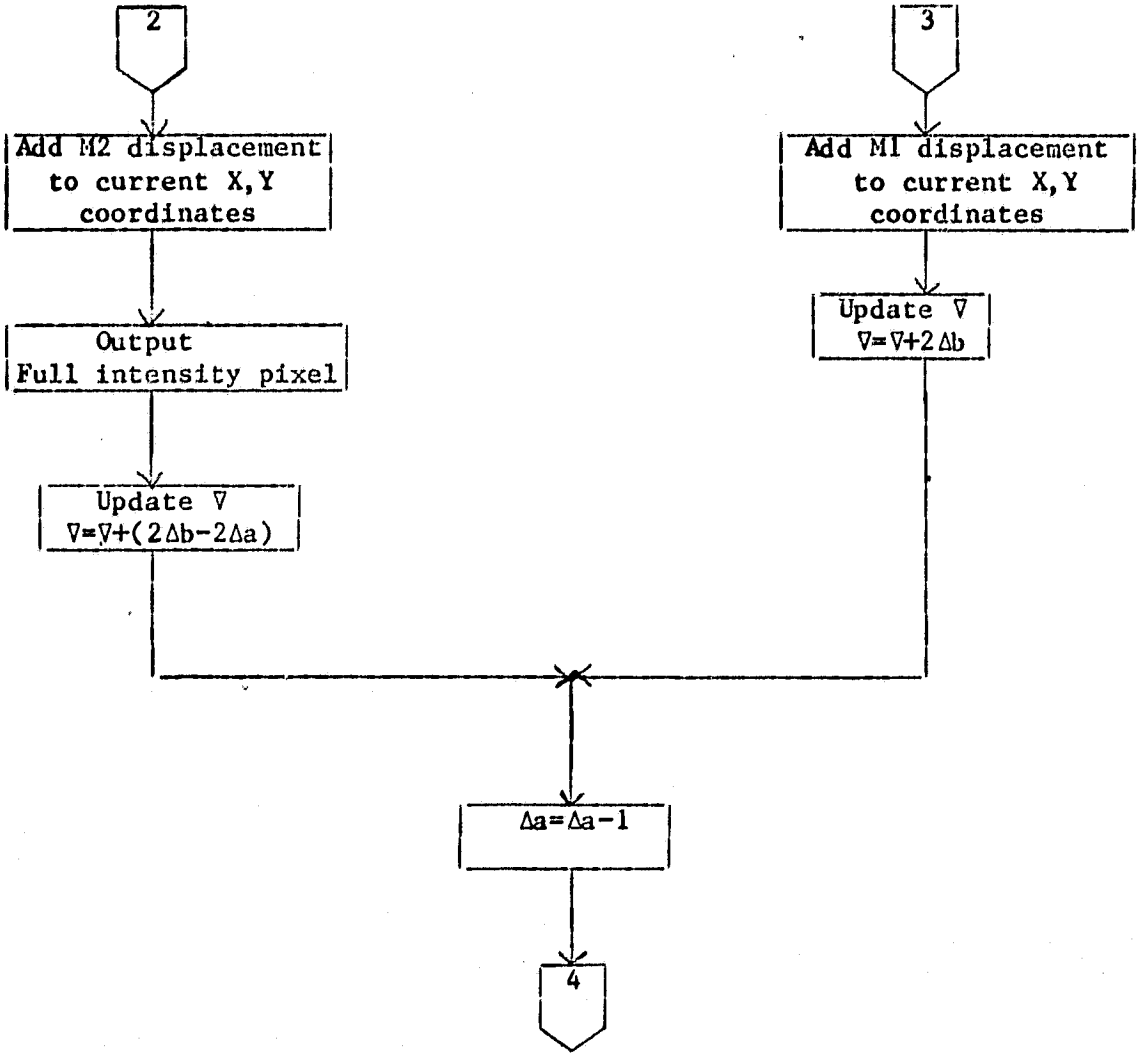


Fig. V.8 (Continued)

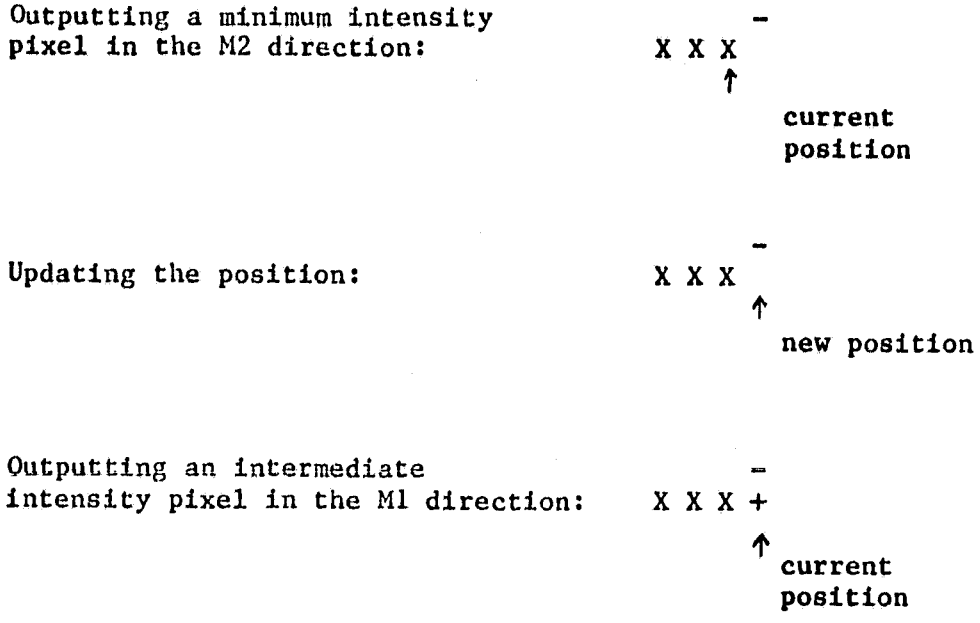


Fig. V.9 Sequence involved in outputting a pixel pair in the first half of the transition region.

If del is greater than or equal to ANTI1, then the second half of the transition region has been entered. Here, the same sequence of operations takes place as in the first half except that the first pixel output (the one in the M2 direction) has an intermediate intensity, and the second pixel output (the one in the M1 direction) has a minimum intensity.

Following the output of a pixel pair in the transition region, del is updated according to the equation

$$V_{i+1} = V_i + 2\Delta b$$

and the M1 loop is exited.

Following the execution of the M1 or M2 loop, the Δa count is decremented. If Δa is not zero, the appropriate move loop is branched to. When the count reaches zero, the line to be generated is complete, and execution of the antialiasing algorithm is finished.

Although the implementation just presented is both fast and visually effective, an understanding of the concepts and constraints for the basic algorithm will permit custom implementations. This particular approach was developed for use in aeronautical cockpit displays where images must move in real-time and be acceptable to pilots. One guiding concept to be kept in mind is: the addition of instructions to the program sections executed prior to line generation will have a much smaller impact on the speed performance than instructions inserted into the move loops. The first class of added instructions would be executed once per line while the second set would be executed once per pixel (a large difference).

V.2. THE ANTIALIASING MICROCODE

Introduction

The effort on this project in the past six months has been directed at implementing the antialiasing algorithm in microcode so that it could be run on an Ikonas graphics system.

The Ikonas is a versatile, real-time graphics computer. By implementing the algorithm in the Ikonas' native microcode, the real-time potential of the algorithm could be verified and quantified. In particular, comparisons to the real-time version of Bresenham's algorithm, also available in microcode, could be made to determine the extent of the speed penalty entailed by antialiasing with this technique.

Note that it is not the intention of this document to offer instruction in reading, and/or writing Ikonas microcode. Rather, this section will provide a discussion of the salient issues involved in translating the FORTRAN version of the algorithm into microcode. Also presented is adequate information to modify the code in order to facilitate its use at other installations. Available Ikonas documentation, cited in the bibliography, would enable complete understanding of the microcode mnemonics as well as the concepts involved in writing microcode.

V.2.1 Microcode Development Hardware and Software

Not until the end of this project did NCSU own an Ikonas RDS-3000, (the original NCSU-owned Ikonas was the prototype model). Consequently, it was necessary to secure the use of an Ikonas to test the microcoded antialiasing algorithm. Permission was received from Jorge Montoya at Research Triangle Institute (RTI) to test the microcode on the Ikonas systems at RTI.

There are two Ikonas RDS-3000 computers at RTI. One belongs to NASA, and the other belongs to RTI. Either one may be connected to RTI's PDP 11/60 which serves as a host for the Ikonas.

The microcode source files were created on NCSU's VAX 11/780 and transferred by tape to RTI's 11/60. There the microcode files were processed by the microcode assembler IKASM4.

As no microcode development work had previously been done at RTI, some modifications had to be made to the microcode development software in order for them to function properly on the RTI system. These modifications included adding a simple breakpoint capability to a microcode debugger, overlaying the same debugger so as to reduce the size of its memory resident task image, and modifying the microcode assembler to permit the use of a greater number of mnemonics and labels.

V.2.2 Microcode Overview

Located in Appendix B are listings of the antialiasing microcode along with the standard Ikonas-supplied microcode implementation of Bresenham's algorithm, labeled VECTLR (VECT stands for vector, LR for low resolution).

The latter appears first in the appendix. It is provided so that the similarities between the standard line drawing microcode and the antialiasing microcode can be readily seen. For example, the microcode that establishes the octant where line drawing is to take place is virtually unchanged from VECTLR. This code is basically all that is common to both algorithms.

Ikonas has an intermediate-level display language called the Ikonas Display Language (IDL). All Ikonas-supplied microcode is accessible via the IDL dispatcher which passes parameters to called microcode functions.

As the display generators for the NASA supported work on cockpit displays at RTI are all written in IDL, and since virtually all display applications are likely to be written in IDL, the antialiasing microcode was made fully IDL compatible. The microcode can thus be invoked, as can any other Ikonas-supplied microcode, from normal IDL programs. The first microcode to be discussed is that which implements the most basic form of the antialiasing algorithm. It is labeled VECTAA1 and is the second listing in the Microcode appendix. AA stands for antialiasing; 1 indicates that this was the first version of the algorithm implemented.

VECTAA1 was designed to implement, in microcode, the FORTRAN program DRAW1 in the most straight-forward way possible. In other words, VECTAA1 was written so that clearly delineated lines of microcode would correspond directly to lines of FORTRAN code that have the same function. By attempting to stick to the structure of the FORTRAN code and by using abundant internal documentation, understanding the essence of the microcode should be relatively simple for anyone who already understands what is going on in the FORTRAN program, and who has the FORTRAN listing handy for comparison with the microcode. Note that the microcode that establishes the octant is taken from the standard microcode, VECTLR, and thus may not correspond exactly to the FORTRAN. Also, the microcode that implements IDL related functions has, of course, no FORTRAN equivalent.

One feature present in the FORTRAN program DRAW1 that was not included in VECTAA1 is the OR'ing of pixels that are to be output in the transition region with the pixel data already present at the pixel location being written to in the frame buffer.

This feature was not included for two important reasons. The first is that in the typical mode of operation, a painter's algorithm is used where the

foreground lines, those that write over other lines at line intersections, are drawn last. Using such an approach, the OR'ing operating becomes superfluous.

The second reason is that a time consuming read from the frame buffer is required for each such OR operation. As speed is of the essence in real-time applications, the OR operation was left out of VECTAA1. However, the OR operation was included in a second version of the microcode, VECTAA2. Thus VECTAA2 represents a faithful microcode reproduction of all the functionality of DRAW1. VECTAA2 will be discussed later.

The following subsections describe the microcodes input data requirements and its output data format. The microcode is then analyzed from the point of view of program flow.

V.2.2.1 Input to the Microcode

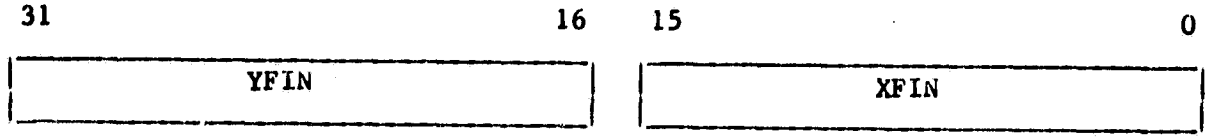
All versions of the microcode are passed a single parameter by the calling IDL program. This parameter specifies the location of a vertex list in memory. A vertex list contains the cartesian coordinates of any number of points to be drawn to or moved to.

Also specified with these coordinates are flag bits. One such flag bit indicates whether the coordinates specify a point to be moved to or a point to be drawn to. The last vertex in the list is flagged with an end bit. A sample vertex list is given in Figure V.10.

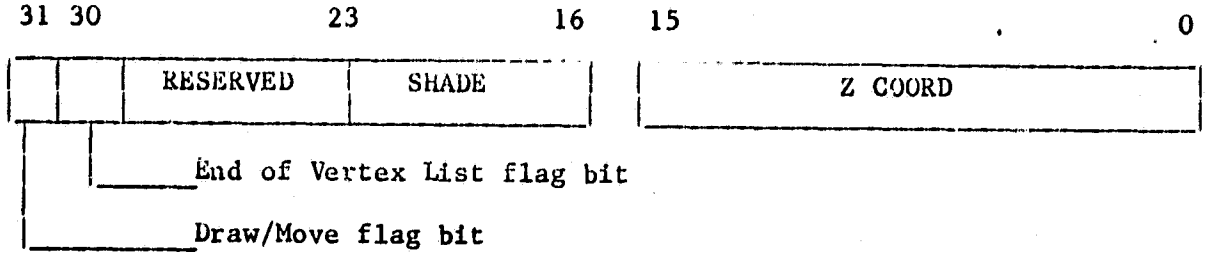
Because a vertex list can be of arbitrary length, the microcode can cause any number of lines to be drawn to the screen by referencing a single vertex list.

The vertex list format used here is identical to the one used by the standard routine VECTLR. However, there is one field in the vertex list that

Vertex Word 1:



Vertex Word 2:



YFIN = y final = y coordinate of point to draw/move to.

XFIN = X final = X coordinate of point to draw/move to.

Fig. V.10 Vertex List Format

is used by VECTLR and is not used by the antialiasing microcode. The antialiasing microcode routines do not read in a shade value from the vertex list. VECTLR uses this 8-bit shade information to generate any of 256 pseudocolored lines.

The ability to draw lines of any color or shade was not incorporated into the antialiasing routines for several reasons. One reason was that the primary goal was to demonstrate the efficacy of the antialiasing algorithm in general and not to develop a line drawing routine fit for a specific application. Another reason was that the FORTRAN prototype for the microcode, DRAW1, contained no provision for drawing lines of any color on demand. Its operation was strictly monochrome. The final reason was that time simply didn't permit the incorporation of this feature into the microcode. Had there been more time, this feature could have been included.

Note that though the antialiasing microcode does not utilize the line shade information specified with the line's coordinates in the vertex list, there are, nevertheless, ways to generate multi-colored lines using these programs. These techniques are discussed in the next section.

V.2.2.2 Output From the Microcode

To understand how the microcode formats its output, it is first necessary to review how the video output path was configured.

The NASA Ikonas at RTI has a frame buffer which is 24 bits deep at each pixel location. The RTI frame buffer, by contrast, is 16 bits deep at each pixel location. Regardless of which Ikonas was running the microcode, only 8 bits of depth per pixel was required.

This is because the Ikonas' frame buffer controller and cross-bar switches were set to "pseudocolor red" operation rather than "full-color" operation.

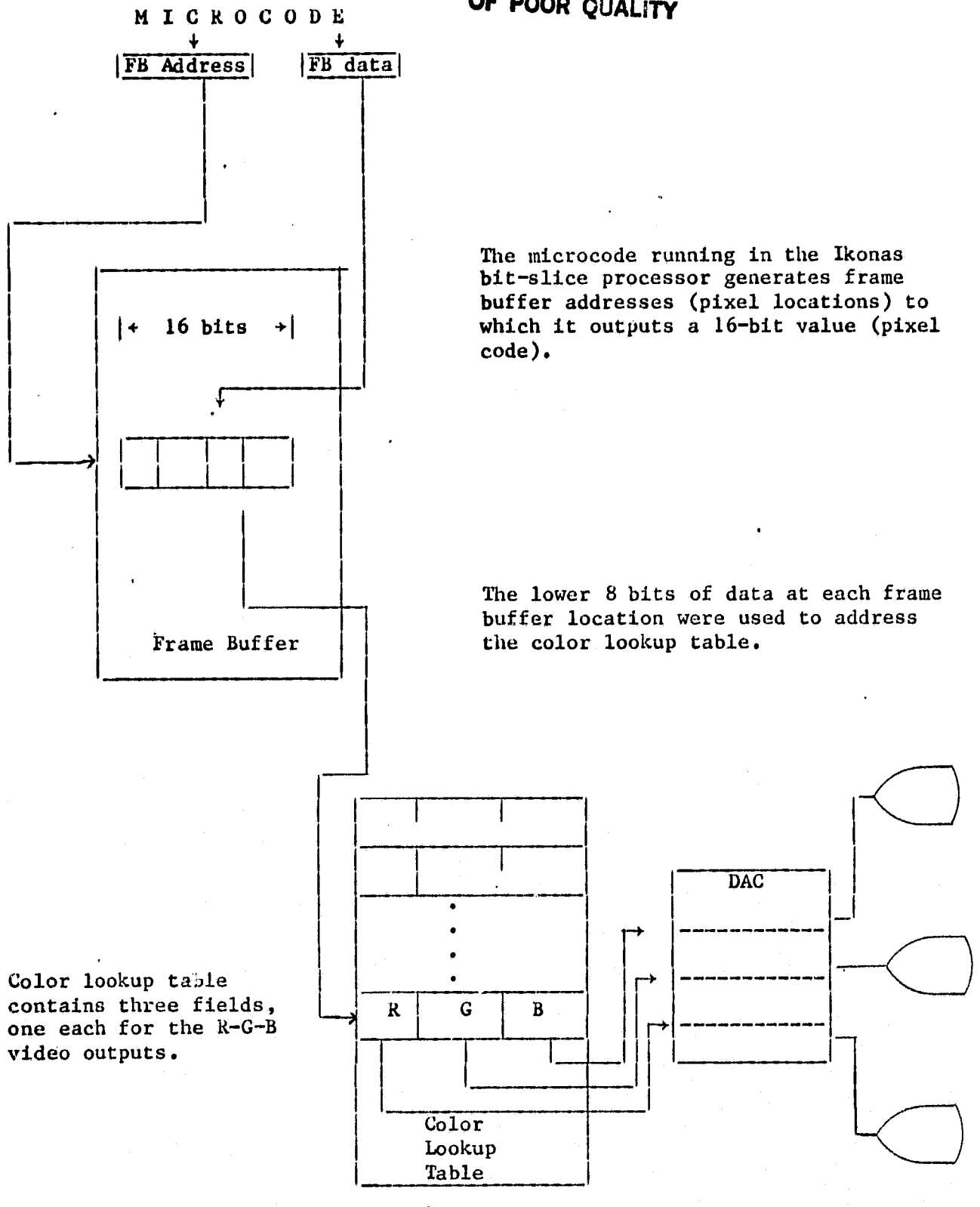
In pseudocolor red mode, the first 8 bits in each frame buffer location (i.e. the 8 bits that would be sent directly to the red channel's DAC's in full-color mode) are used to address a 256 place color lookup (LU) table. Each location in the LU table contains 24 bits of data: 8 bits for red shading, 8 bits for green shading, and 8 for blue shading. It is thus the output of the color lookup table that is sent to the R-G-B DAC's, (see Figure V.11).

To function as planned then, the microcode must, for each pixel it outputs, place a value in the frame buffer which will be used to address a location in the lookup table where the proper intensity information is stored. Of course, this requires that the lookup table be loaded with the proper values prior to executing the microcode. This task of initializing the lookup table is performed by the IDL program which calls the antialiasing microcode.

For example, if a full intensity white pixel was to be drawn, location 256, say, in the lookup table might be used to store the 24 one's that would represent full white in digital form: 8 bits for full red, 8 bits for full blue, and 8 bits for full green. So to initialize the lookup table for full white, the IDL program would store FFF FFF (full intensity white) in location FF (256).

Later, during line drawing, if the microcode was required to output a full intensity white pixel, it would write FF to that pixel's location in the frame buffer. In fact, the microcode happens to write 16 bits to the frame buffer for each pixel output, not just the 8 bits required. Only the lower eight bits are used to address the lookup table, however; the other 8 bits are ignored by the hardware controlling the frame buffer.

It was noted previously that this microcode implements only monochrome line drawing; i.e. any shade or color specified with the line's coordinates in



The microcode running in the Ikonas bit-slice processor generates frame buffer addresses (pixel locations) to which it outputs a 16-bit value (pixel code).

The lower 8 bits of data at each frame buffer location were used to address the color lookup table.

Color lookup table contains three fields, one each for the R-G-B video outputs.

Fig. V.11 Simplified data flow diagram illustrating the microcode output and video output paths.

the vertex list are ignored by the microcode. Yet it was also mentioned that there were ways of drawing lines of any color.

The microcode outputs only color independent values into the frame buffer. These are used to reference the LU table where the full, intermediate, and minimum intensity values are stored. Obviously, since the lookup table is loaded by the user (from an IDL program), an antialiased line of any given color may be drawn by loading the locations corresponding to full, intermediate, and minimum intensities with the corresponding intensity values for the desired color.

The following table (Table V.2.2-1) lists the actual values output by the microcode. The selection of red as the line color was arbitrary as were the address in the lookup table used to hold the 3 intensity values.

First of all, note that the data in the lookup table is 10 bits long, rather than 8 bits long as would be expected for generating the color red based on the previous discussion. As it happens, 10-bit DAC's are used on the video output and so 30 bits (10x3) as opposed to 24 bits (8x3) must be stored at each LU table location. This doesn't really affect the previous discussion, though, as the lower 2 bits of the 10 bit fields are simply set to zero, (e.g., 377→1774).

Now for the sake of clarity, let's return to discussing the LU table values as 3 fields of 8 bits each, with the least significant 8 bits used for red.

The full intensity red is octal 377 (255 decimal) -- 8 bits into the red DAC are on. This is as expected, but the 2/3 and 1/3 intensity values are not 2/3x255 = 252 (octal) and 1/3x255 = 125 (octal) as one might expect. This is due to the fact that the gamma correction curve for red is not linear. So to

Pixel Intensity Output by Microcode	Actual Pixel Code Output by Microcode	Location in LU Table addressed by pixel code	10 bit value stored at each LU location by IDL program	Corresponding 8 bit value	Resulting Color and Intensity level
FULL	$177777_8 = FF\underline{F}\underline{F}_{16}$	$FF_{16} = 255_{10}$	1774_8	$377_8 = 255_{10}$	Full intensity red
MED (~2/3 intensity)	$114631_8 = 99\underline{9}\underline{9}_{16}$	$99_{16} = 153_{10}$	1522_8	$324_8 = 212_{10}$	Intermediate intensity red
MIN (~1/3 intensity)	$042104_8 = 44\underline{4}\underline{4}_{16}$	$44_{16} = 68_{10}$	1250_8	$252_8 = 170_{10}$	Minimum intensity red

Table V.2.2-1 Microcode output and color lookup table values for drawing antialiased red lines. The above values were used in the IDL programs AA1GRID2 and AA2GRID2 which drew antialiased test patterns with VECTAA1 and VECTAA2 respectively.

perceive the fractional intensities as 2/3 and 1/3, the values sent to the DAC's must be significantly greater than 2/3·(255) and 1/3·(255).

The values selected for the fractional intensities were not derived from a gamma correction curve but were arrived at experimentally. Thus they are only an approximation to the 2/3 and 1/3 intensity values they are intended to represent.

V.2.3 Analysis of the Microcode

V.2.3.1. VECTAA1

A high-level analysis of the microcode program flow is next discussed. Flowcharts are presented to aid in understanding the concepts involved. A high-level flowchart for the antialiasing routine VECTAA1 is presented in Figure V.12.

The first operation is to get the vertex pointer. Recall that the vertex pointer is the parameter passed to the microcode from the calling IDL program. The pointer contains the address in memory of the vertex list -- the list of positions to be moved to or drawn to.

After the vertex pointer has been passed in, the microcode uses it to read in the next vertex from memory.

Following this, several set-up operations are performed. They are thus named because they set up the contents of certain registers prior to line drawing. In these set-up operations, the register containing the old starting coordinates is loaded with the old ending coordinates (of a move or draw), and the register containing the old ending point is loaded with the newly read in vertex. Note that this updating of the starting and ending points of a line is all that is required for a move operation.*

Another set-up operation involves changing the input vertex from screen coordinates to hardware coordinates. In the screen coordinate system, the lower left corner of the screen is taken to be the origin. All user specified vertices are specified in screen coordinates. By contrast, the hardware takes the upper left hand corner of the screen to be the origin, thus necessitating

*Note: A 'move' in this context refers to the repositioning of the starting point of a line to be drawn. It does not refer to the M1 or M2 moves made by the line drawing routine in the course of drawing a line.

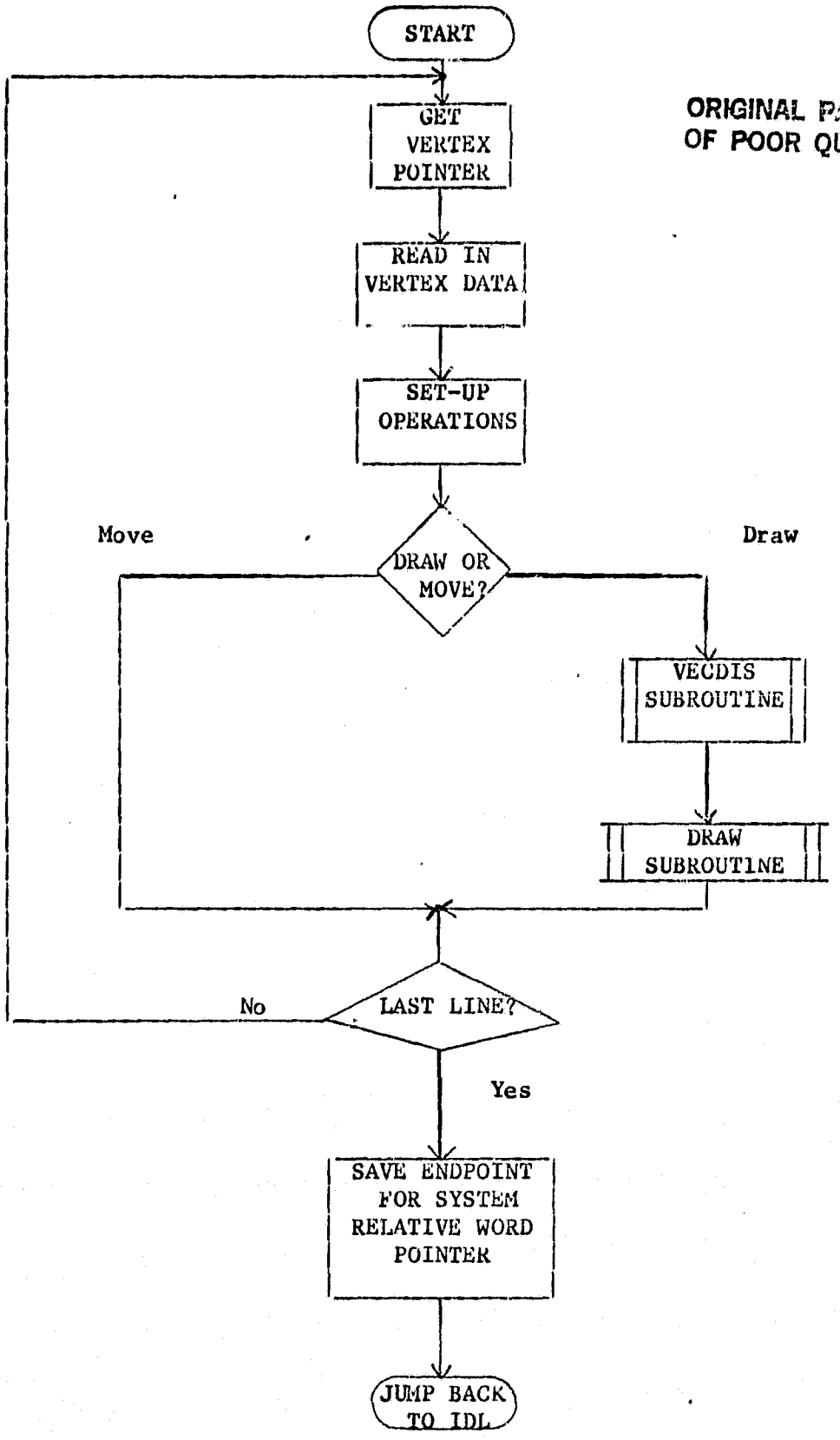


Fig. V.12 High-level flowchart for the antialiasing microcode

the translation from user oriented screen coordinates to the hardware coordinate system.

Following the set-up operations, the most significant bit in the vertex word pair that was read in from memory is examined to determine whether a move or a draw to the x,y position is indicated, (see Figure V.10 for vertex word pair format).

If bit 31 of the second vertex word is set, a draw is indicated. As seen in the high-level flowchart, a positive test of the draw bit causes two microcode subroutines to be executed. The first, labeled VECDIS, establishes the octant of the line to be drawn and calculates the normal Bresenham vector displacements - Δa, Δb, Δx, and Δy. This subroutine is unchanged from the microcode which implements the standard Bresenham's algorithm (VECTLR).

The second subroutine invoked is labeled DRAW and, as the name implies, this subroutine actually draws the antialiased line. More will be said about DRAW shortly.

On return from the DRAW subroutine, the program merges once again and the end of vertex list flag bit, bit 30 of the second vertex word, is checked to see if the end of the vertex list has been reached.

If the end of the vertex list has not been reached, then more moves and/or draws remain to be done. The program branches back to get the next vertex word pair.

If the end of the vertex list has been reached, the last vertex is written out to a system reserved memory location where it can be used for any subsequent graphics operations that are positioned relative to the last endpoint (hence the name of this location --the relative word pointer).

Finally, a jump to the IDL system routine IDLEXIT returns control to the IDL dispatcher. This is an unconditional jump and not a return from

subroutine command because the microcode of VECTAA1 is jumped to and not called as a subroutine.

Before discussing DRAW in more detail, a further note on the move branch of the flowchart is in order. When bit 31 was checked and found clear, a move to the next vertex is indicated. Though the move branch of the conditional has no operations in it, the required operations for a move have already been performed by the previously mentioned set-up operations.

The flowchart for the microcode subroutine DRAW is given in Figure V.13. Comparing this flowchart with the listing for the FORTRAN subroutine DRAW1 demonstrates the fidelity with which the program structure of the FORTRAN routine was maintained by the microcode.

As the FORTRAN program has already been extensively discussed, no further explanation will be presented of the basically self-explanatory microcode flowchart of Figure V.13.

A more in depth understanding of the microcode requires some familiarity with the Ikonas architecture and the microcode mnemonics. Documents providing this type of information are referenced in the bibliography. Using these documents and the microcode's own internal documentation, the microcode can be readily understood on a line-by-line basis.

V.2.3.2 VECTAA2 and Pixel OR'ing

One feature of the FORTRAN subroutine DRAW1 that was not implemented in VECTAA1 was the ability to logically OR the pixel value to be output with the pixel value already at the location to be written to. The result of this OR operation is that the pixel with the greatest intensity is the one that is written out to the frame buffer.

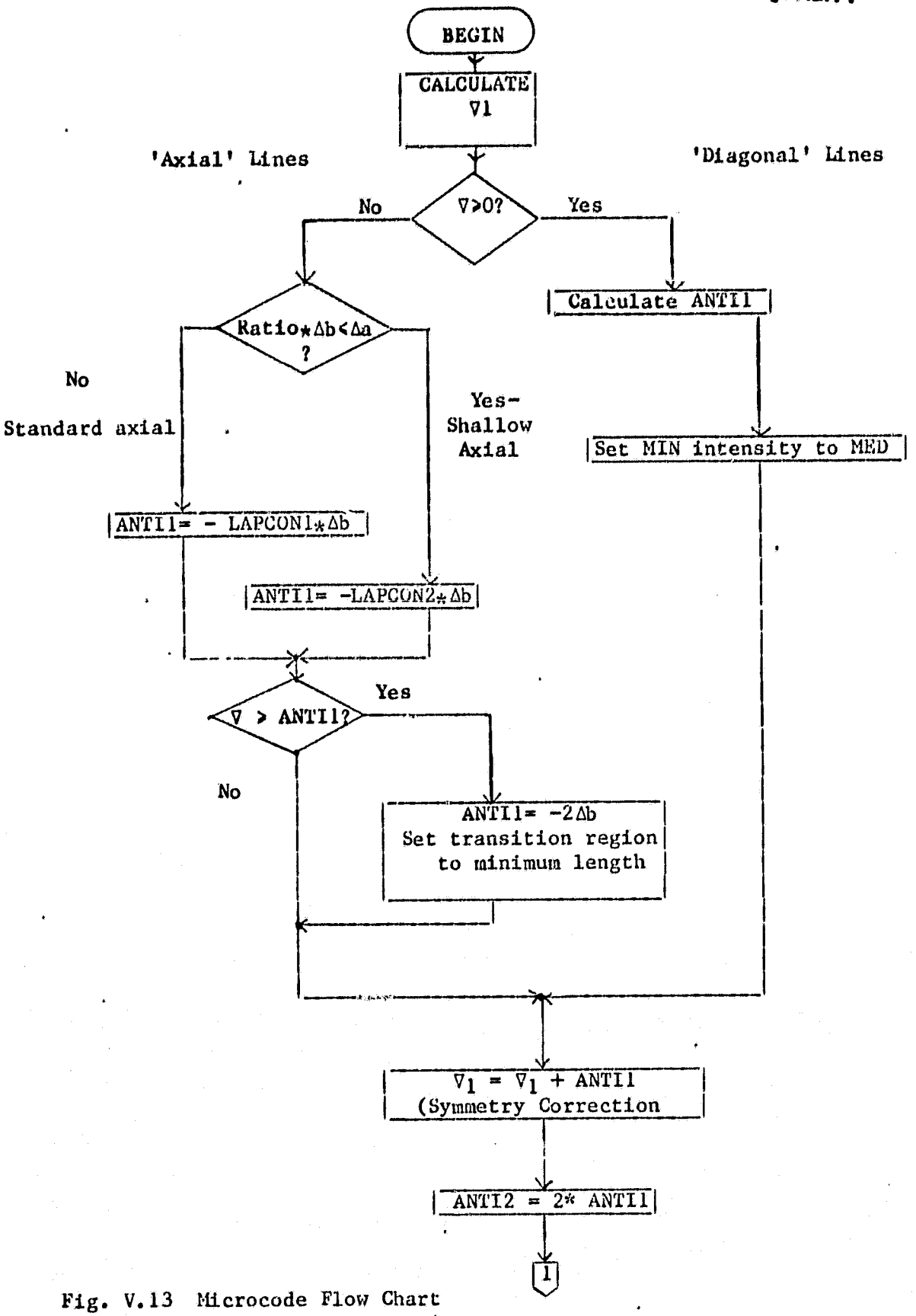


Fig. V.13 Microcode Flow Chart

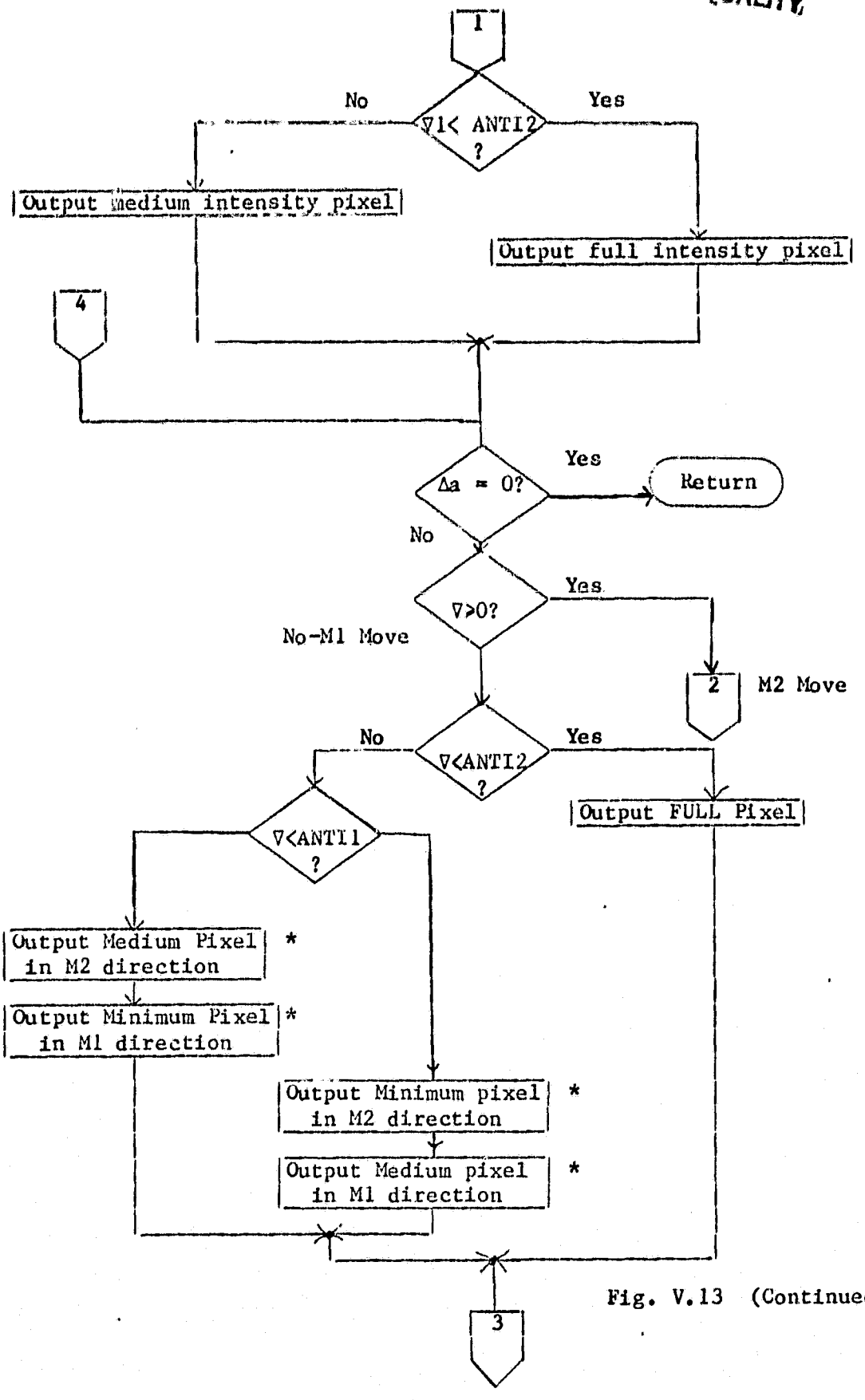


Fig. V.13 (Continued)

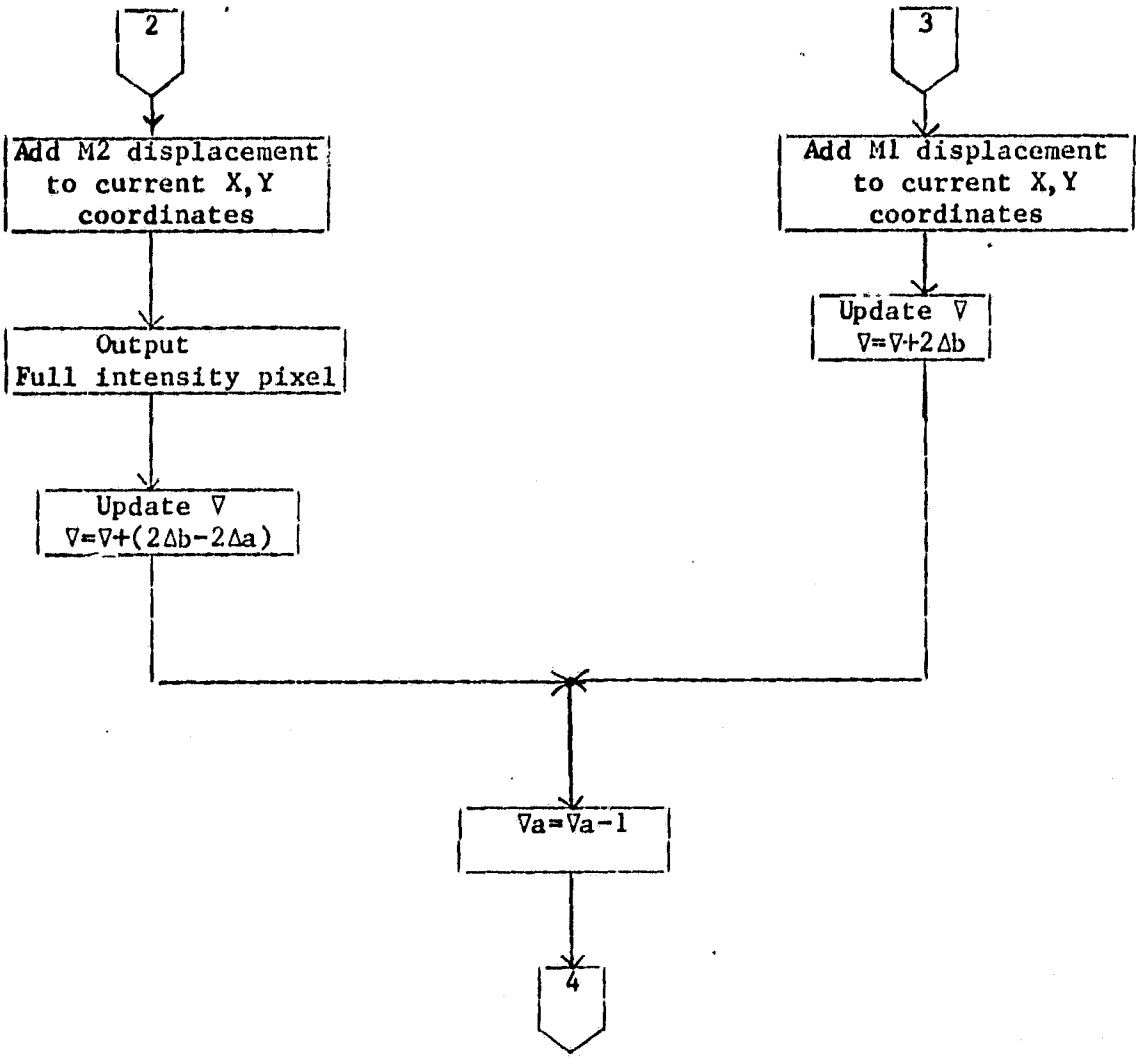


Fig V.13 (Continued)

The way this OR'ing is implemented is that whenever a transition region pixel, (which is either medium or minimum intensity) is to be output, the pixel value at the addressed location is read in from the frame buffer. If the read-in pixel has a greater intensity than the transition region pixel (as determined by the OR of the two), then it is the read-in pixel which is written back to the frame buffer.

The OR operation is not used when full intensity pixels are to be written out. Full intensity pixels overwrite whatever is already in the frame buffer.

The practical effect of all this is evident when an antialiased line is being drawn over extant lines in the frame buffer. If the antialiased lines transition region crosses the full intensity pixels of another line, the full intensity pixels will win out.

Figure V.14 demonstrates the difference between a line drawn with VECTAA1 (without OR'ing) and VECTAA2 (with OR'ing).

Red letters - Indicate full intensity line already in the frame buffer.
Black letters - Indicate antialiased line being drawn into the frame buffer.

Fig. V.14 Illustration of effect of pixel OR'ing by comparing lines drawn by VECTAA1 with lines drawn by VECTAA2.

VECTAA2 produces a somewhat more visually acceptable display because the full intensity line already in the frame buffer is not "interrupted" by the two diminished intensity pixels from the transition region of the other line.

As stated in section V.1.2, there is a possibility that the transition region of the line being drawn will overlap another line's transition region. Under such circumstances, the OR'ing of the two diminished intensity pixels could produce a spurious full intensity pixel.

Also, in the case of this pseudocolor implementation, if the medium and minimum intensity pixel codes are not selected so that they OR to a full intensity pixel code, the OR of a medium and minimum intensity pixel could result in a pixel code which references an uninitialized location in the color lookup table. The pixel intensity output to the display would thus be unpredictable. This problem could be avoided by proper selection of pixel codes.

There would still be the problem of getting a spurious full intensity pixel code output output as the result of the OR. As was stated earlier, the probability of a line intersection of this type is sufficiently low, that the occasional anomally that results from the OR under these conditions is visually acceptable.

The drawback to VECTAA2 is that since each pixel OR takes several additional instructions, a substantial speed penalty is incurred.

The frame buffer read operation alone adds two instructions to the inner most loop of the microcode.

The magnitude of the speed reduction depends on the line being drawn. Lines drawn with nothing but full intensity pixels output, vertical and horizontal lines, will be drawn at the same rate as with VECTAA1. This is because the OR procedure only occurs in transition regions, and transition regions are not present in these types of lines.

On the other hand, for lines with large transition regions, such as shallow axial lines, the speed penalty incurred by OR'ing each pixel in the transition region is substantial.

Numbers relating the exact extent of the speed penalty are presented in Sec. V.2.5 under testing.

Those blocks of the flowchart in Figure V.13 that have stars beside them are those that are changed in VECTAA2. Figure V.15 shows how these blocks are reproduced for VECTAA2.

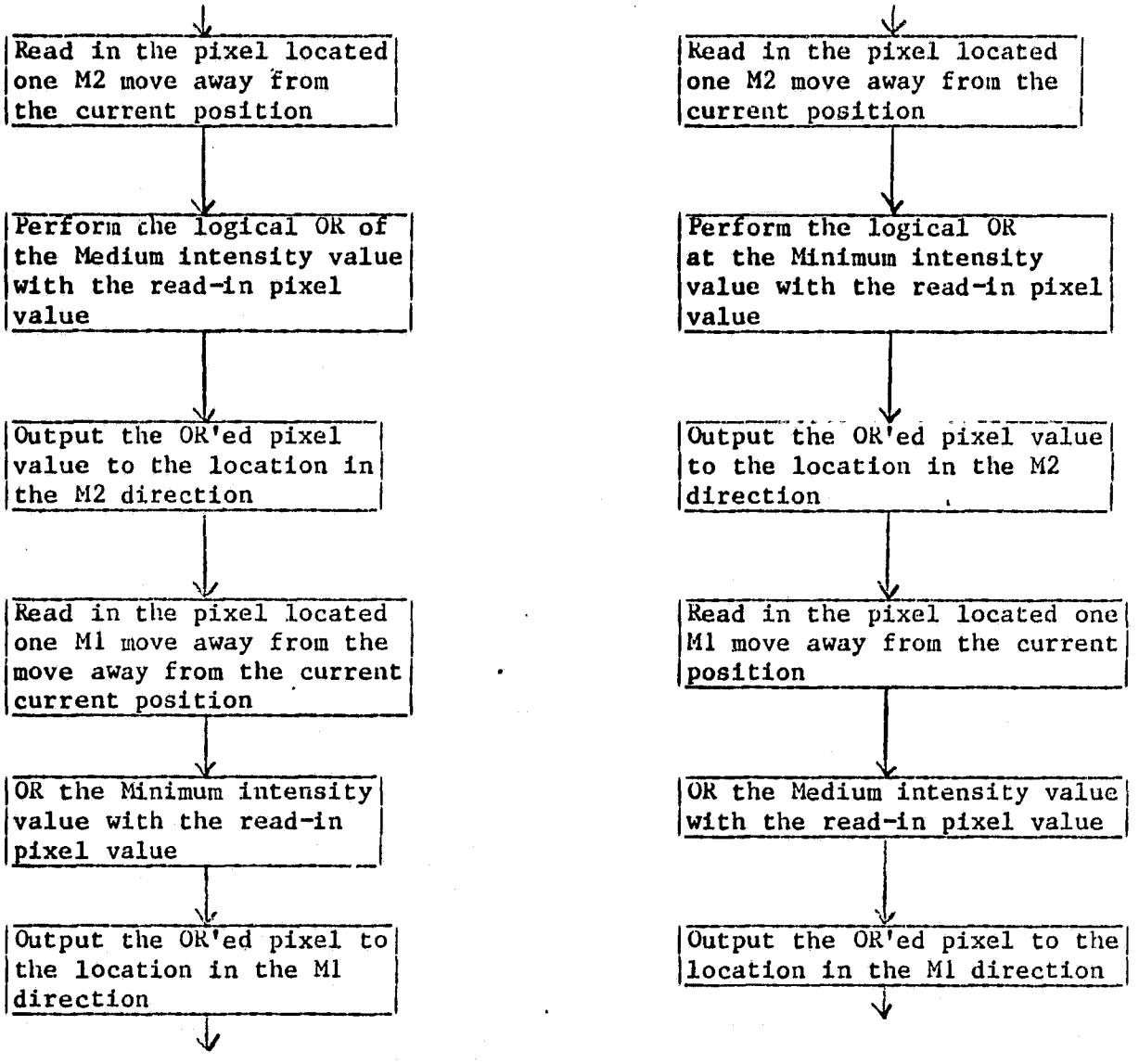


Fig. V.15 VECTAA2 flowchart blocks that are changed from VECTAA1. These blocks depict the transition region pixel OR'ing.

V.2.4 Pipelining and the 'FAST' Microcode

The Ikonas bit-slice processor has the ability to pipeline instruction execution. That is, under some circumstances two different operations may be invoked from a single instruction. In practical terms, this allows ALU operations to be invoked from the same instruction that is testing a condition code, performing a pixel write, or issuing a jump command. This instruction overlap has the obvious potential of condensing the microcode to fewer instructions and thus reducing execution time.

In VECTAA1 and VECTAA2, the full pipelining ability of the Ikonas was not, in some instances, utilized. The reason for this is that writing understandable, and hence easily debugged, microcode is facilitated by having each line of microcode map to a single box in the flowchart. Also, the debugging process was ameliorated because tracking down problems is less of a chore when there are not several operations going on during each line of microcode.

Nevertheless, since speed was the critical goal in this effort, faster versions of the microcode were developed. After VECTAA1 and VECTAA2 had been debugged, versions of the microcode that fully utilized pipelining were created. These programs were labeled VECTAA1F (F for fast). The listings for these programs appear in Appendix B.

Line by line comparison of the first versions of the microcode with the "F" versions show very little difference. In all, only 3 or 4 instructions were saved in the 'F' versions. But the fact that one of these removed instructions occurred in the inner most loop of the microcode DRAW subroutine, resulted in close to a 10% reduction in overall execution time.

V.2.5 Testing

Testing the microcode was split into two stages. The first involved creating data sets (vertex lists) that would test the the program's ability to draw all types of lines correctly. The second stage consisted of getting a measure of the rate at which the program could draw lines. This rate, of course, has an important impact on the real-time applicability of the algorithm/microcode.

V.2.5.1 Test Pattern Generation

Figure V.16 shows a test pattern used for the 1st stage of testing. This pattern consists of 40 lines drawn radially outward from the center of the display.

Ending point coordinates were chosen so that each type of line -- diagonal, axial, and shallow axial -- would be drawn in each octant of the display.

A close-up view of a portion of this pattern is shown in Figure V.17. Full intensity pixels appear red in this picture. To aid in discerning the transition regions, the transition region's diminished intensity pixels have been pseudocolored green for intermediate intensity and blue for minimum intensity.

As can be seen, the center line in the picture has no transition regions because it is a purely horizontal line. The pair of lines surrounding the center line are shallow axial lines. These lines have extended transition regions to enhance their appearance. The outermost pair in the picture consists of standard axial lines. These lines have the normal length transition region for axial lines.

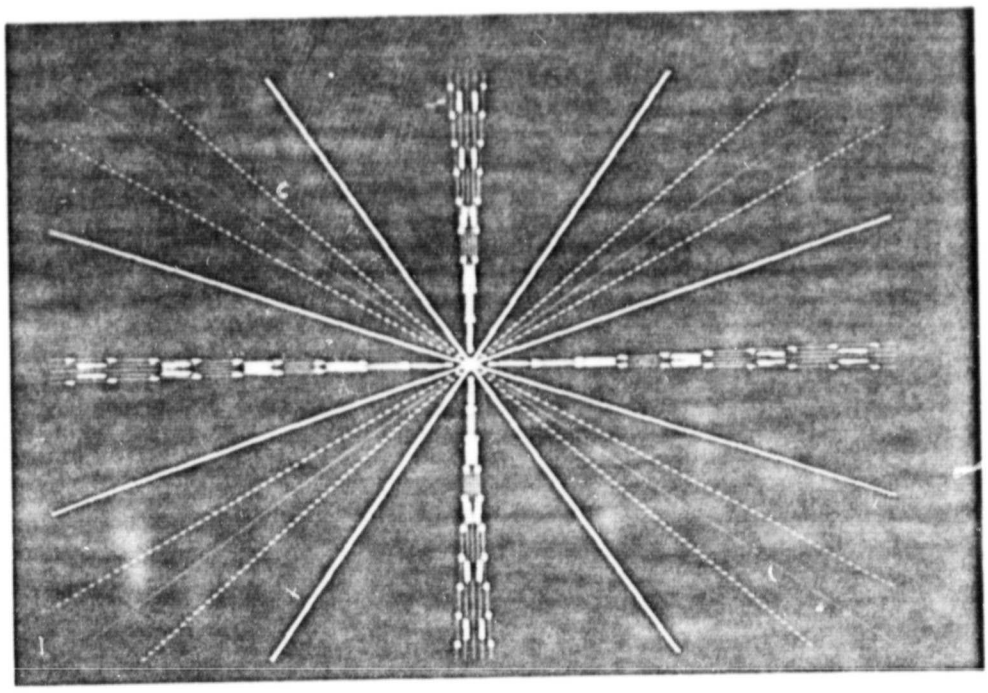


Figure V.16 Test Pattern

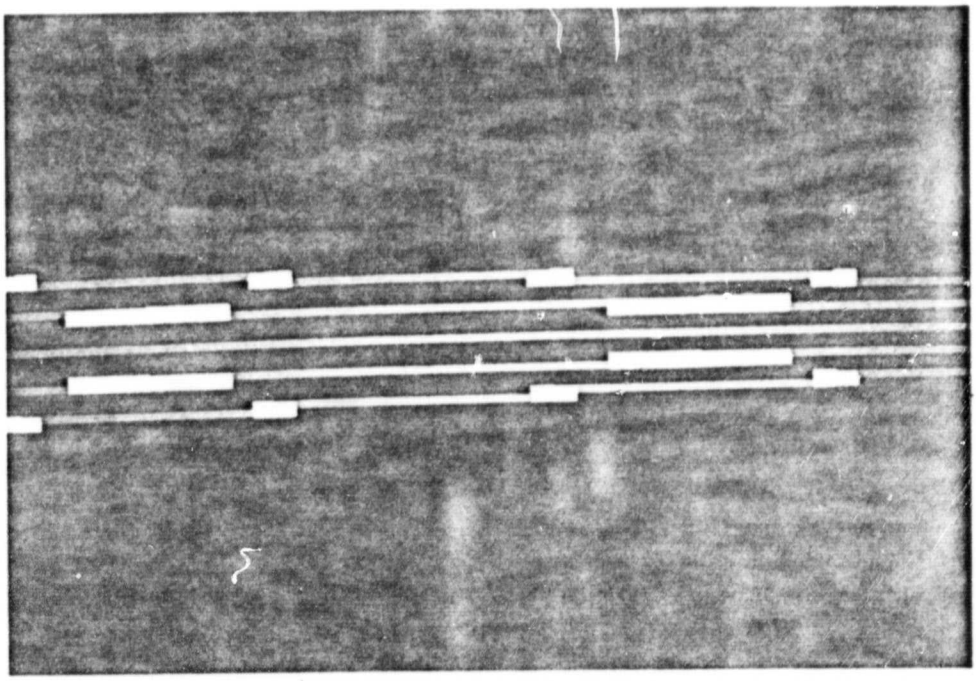
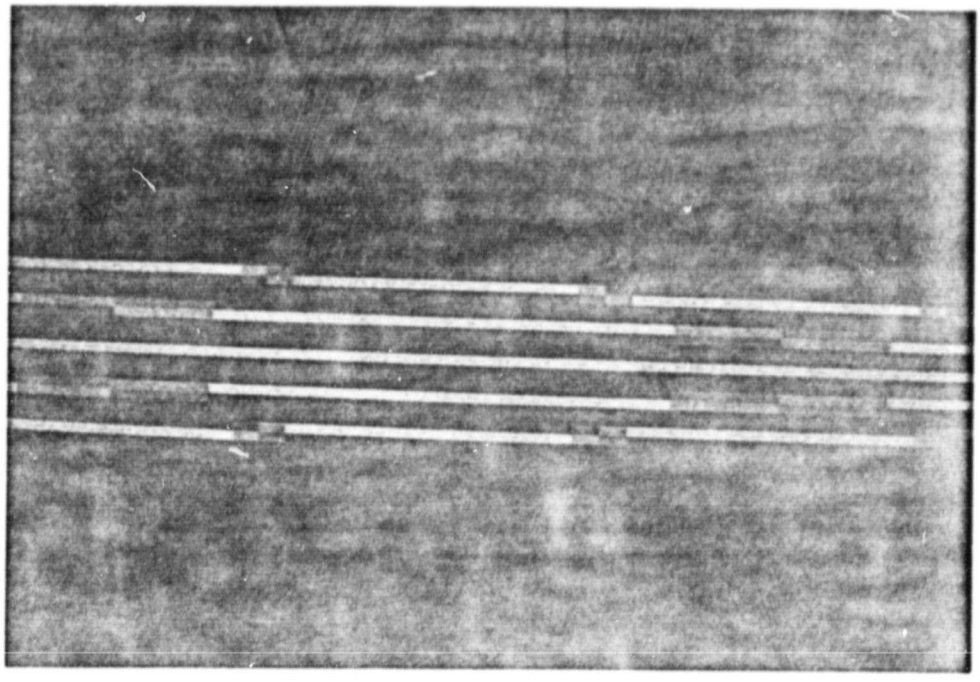
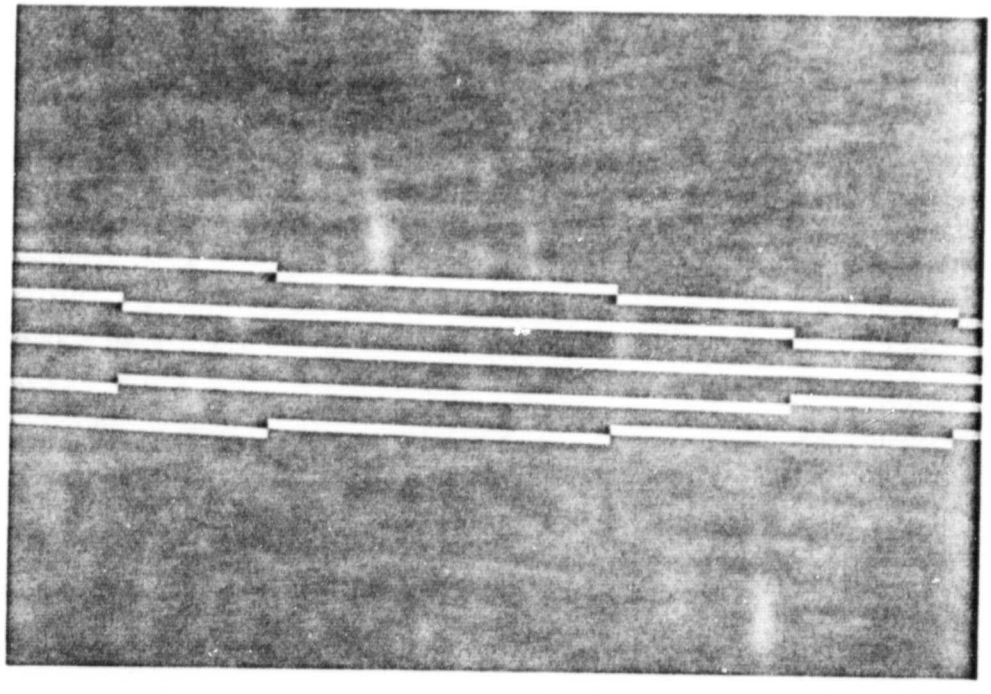


Figure V.17 Test Pattern Zoomed

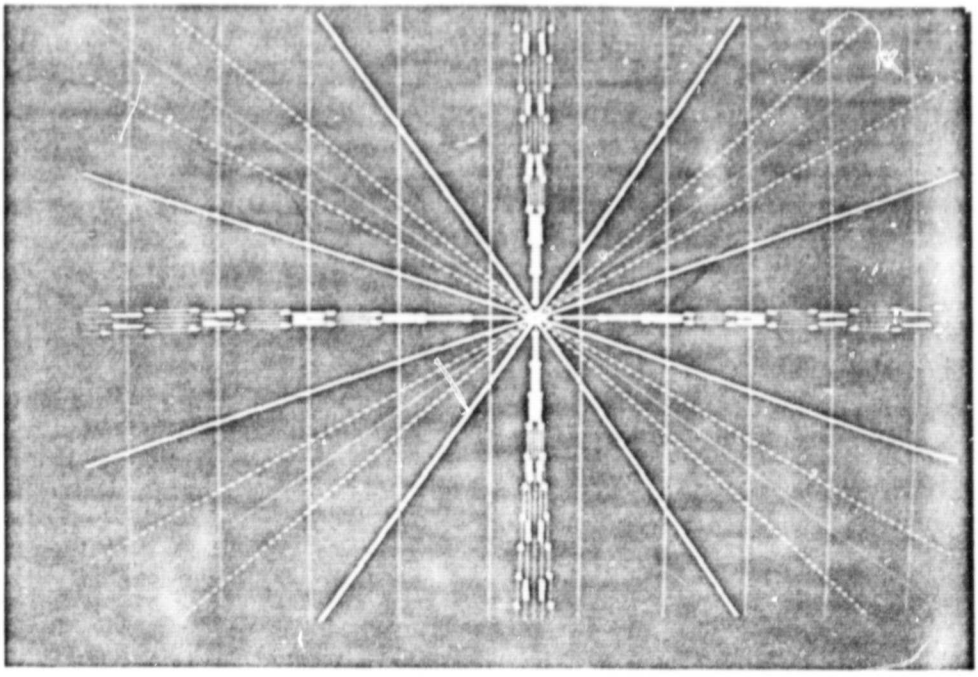
Figure V.18



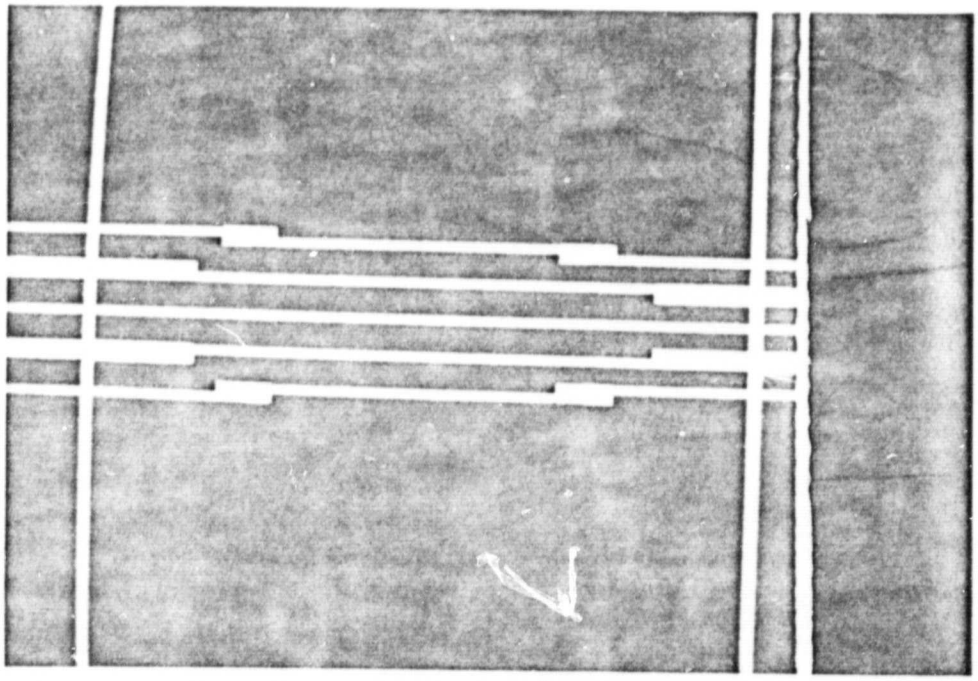
(a)



(b)



(a)



(b)

Figure V.19

Figure V.18(a) shows the same area as figure V.17, but this time with the transition region's pixels properly shaded with intermediate and minimum intensity red pixels. For comparison, figure V.18(b) shows this portion of the test pattern drawn with the standard Bresenham routine. The smoothing effect of the antialiasing process is clearly apparent in this a-b comparison.

Note that the antialiased lines in figure V.18(a) do not really appear very "smooth" because we are viewing these lines at extreme magnification.

Thus, this first pattern was used to demonstrate that both VECTAA1 and VECTAA2 drew lines properly in all octants.

A second test was used to show that the transition region pixeling OR'ing was properly performed by VECTAA2. The first step in this test involved placing a set of vertical full intensity lines in the frame buffer. The test pattern of Figure V.16 was then drawn by VECTAA2 over these vertical lines. The desired result was that whenever a transition region pixel (either blue or green) was to be output over a full intensity red pixel in one of the vertical lines, the red pixel was to remain unchanged. VECTAA2's pixel OR'ing worked properly and this desired result was achieved. Figures V.19(a), (b), show respectively the test pattern overlaid on these vertical red lines, and a closeup of a transition region showing how the vertical full intensity line was not written over by the transition region.

V.2.5.2 Timing Tests

Measuring the antialiasing routines' speed performance constituted the second phase of testing. Rather than try to establish an absolute scale by which to measure this performance, such as pixels output per second, a relative measure of the routine's performance against the standard Bresenham routine was decided upon.

Table V.2.5-1 lists the normalized timings of VECTAA1F and VECTAA2F over all classes of lines. Because of the normalization, a timing of 2.0 means that the line took twice as long to draw with the antialiased routine as with the standard Bresenham routine.

Relative timing tests were made by having both the antialiasing microcode and the standard Bresenham microcode draw identical lines of the same length (500 pixels) a given number of times (each line was drawn 60,000 times). A stopwatch was used to measure the time it took each routine to draw the 60,000 replications of the given line. Several trials for each type of line (axial, diagonal, etc.) were conducted so that the timing results would be reliable.

Looking at the figures for VECTAA1F, the best results were achieved when drawing 45° diagonal lines. These lines are composed entirely of M2 moves. As can be seen from the flowchart (Figure V.13), the program flow path for the M2 move involves fewer instructions than for the M1 move. Thus, a line consisting of only M2 moves will be drawn faster than one which contains M1 moves.

The second best time was turned in by the diagonal type line that makes a 38.65° angle with the horizontal axis. These lines consist of both M2 and M1 moves, but the predominant number of moves is in the M2 direction.

Vertical and horizontal lines appear next in the table. These lines consist exclusively of M1 moves. Because these lines have no transition regions, full intensity pixels are always output by the microcode. From the flowchart it is obvious that of the three paths available in case of an M1 move, the one containing the fewest instructions is the one in which a full intensity pixel is output. This accounts for the fact that these lines are drawn more rapidly than the ones which appear further down in the table.

NORMALIZED TIMING FIGURES
FOR VECTAA1F and VECTAA2F

VECTAA1F:

Line type	Time
45° Diagonal	1.40
38.65° Diagonal	1.67
Vertical	2.02
Horizontal	2.02
Standard Axial	2.10
Shallow Axial	2.13

VECTAA2F:

Line type	Time
45° Diagonal	1.40
Vertical	2.02
Horizontal	2.02
38.65° Diagonal	2.08
Standard Axial	2.36
Shallow Axial	2.80

Table V.2.5-1 Relative timing tests for VECTAA1F and VECTAA2F. Timings were made on a 30 Hz display with the frame buffer controller initialized for 30 Hz update rate.

The next table entries are for axial lines. These lines involve the longest inner loop or path through the microcode. The times for shallow axial lines are slower than for standard axial lines because the transition regions are 4x as long for shallow axial lines as for standard axial lines.

The second part of the table contains the timing figures for VECTAA2F. Comparing the timings of VECTAA1F with VECTAA2F, one notices that the time to draw a 45° diagonal, horizontal or vertical line is the same for the 2 versions of microcode. This is because these types of lines are composed entirely of M2 or M1 moves. They contain no transition regions and hence no pixel OR'ing. All the other tested line types contain transition regions. These lines are drawn slower by VECTAA2F (which performs pixel OR'ing in transition regions) than by VECTAA1F.

The listing of line types in order of decreasing draw rate is much the same for VECTAA2F as for VECTAA1F. The only difference is that with VECTAA2F, the vertical and horizontal (all M1 move) lines are drawn faster than the 38.66° "diagonal" line. This is because the "diagonal" line contains transition regions. With VECTAA1F, the lines were drawn quickly because of their predominance of M2 moves. With, VECTAA2F, these lines are drawn more slowly than the all M1 move lines because the pixel OR'ing operation in VECTAA2F washes out the time saved by the predominance of M2 moves.

V.2.5.3 Real-time Display Testing

As a final test of its real-time feasibility, the pitch bars of the EADI display under development at RTI were drawn with the antialiasing microcode routine. The real-time performance of the EADI display interacting with a NAVION aircraft simulation was not adversely affected by the use of the routine (VECTAA1F), and the pitch bars appeared properly antialiased.

V.3. APPLICATION NOTES

V.3.1 Future Implementation of the Microcode

These microcode implementation notes are presented to aid in the future use of these antialiasing routines.

The microcode as written should, after being assembled with IKASM4, run on any Ikonas RDS-3000 graphics computer.

However, such site dependencies as the number of bit planes in the frame buffer and the configuration of the video output path (full color, pseudo-color, etc.) will effect how the output of the microcode is translated to colors and intensities on the monitor. For this reason, the microcode constants equated to the FULL, MED, and MIN pseudo-ops, would be changed based on how deep the frame buffer was, how the color lookup table was loaded, and so forth.

As stated before, the microcode should run on any current generation Ikonas. It can not be guaranteed, however, that the IDL programs supplied here to illustrate how to call the antialiasing microcode will be valid in the future. A new release of IDL is expected soon from Ikonas. The two versions of IDL may or may not be compatible.

V.3.2 Recent Developments

As this project was being brought to a close, Ikonas announced the imminent release of several new software products.

Among the new releases is an improved version of the Bresenham algorithm microcode which runs about twice as fast as the former Bresenham microcode, VECTLR.

Also soon to be available from Ikonas is microcode to draw variably shaded antialiased lines in real time. The line drawing rate of this microcode (which is, incidently, based on an entirely different algorithm than the microcode discussed here) is said to be approximately equal to the old VECTLR microcode. If this is so, then the Ikonas antialiasing microcode has a significantly higher line drawing rate than the microcode developed at NCSU (which is somewhat slower than VECTLR).

In light of these announcements, the practicality of any future implementation of the NCSU antialiasing microcode must frankly be questioned. Note, however, that the announcement of the Ikonas antialiasing microcode does not "invalidate" the work done on this project. The goal of this effort was to produce a visually effective antialiasing algorithm fast enough to be used in real time. The results of the preceding section on testing make plain the fact that this goal was indeed achieved.

VI. CONCLUSIONS

The worked performed under this research contract has covered an extensive area in computer graphics. The main exphasis has been real-time computer graphics with aeronautical applications.

The result has been both hardware and software which have laid the ground work for continued efforts at the Research Triangle Institute and IKONAS Graphics systems (now Adage). Although, this phase of the research will be finished here at NCSU it is hoped this advanced development will continue elsewhere.

VII. REFERENCES

[1] J. E. Bresenham, "Algorithm for Computer Control of a Digital Plotter," IBM System Journal, Vol. 4, 1965.

[2] J. Barros and H. Fuchs, "Efficient Generation of Smooth Line Drawings on Video Displays," SIGGRAPH '79 Proceedings, pp. 260-269, August 1979.

[3] J. E. Bresenham, "Incremental Line Compaction," IBM System Communications Division, 1980.

APPENDIX A

```
0001            SUBROUTINE DRAWO(STARTX,STARTY,ENDX,ENDY)
               C*****************************************************************
               C
               C       LINE PLOTTING ROUTINE USING STANDARD BRESENHAM ALGORITHM
               C               - THIS SUBROUTINE DOES NOT DO ANY ANTIALIASING AND
               C               ANY COMMENTS PERTAINING TO THAT MAY BE IGNORED, IT
               C               GENERATES ONLY FULL INTENSITY PIXELS (INTENS(1)).
               C
               C*****************************************************************
               C
               C               CALL PARAMETERS
               C
               C       STARTX,STARTY = X & Y COORDINATES OF LINE START POINT
               C       ENDX,ENDY     = X & Y CORRDINATES OF LINE END POINT
               C
               C*****************************************************************
0002            IMPLICIT INTEGER*2(A-Z)
0003            COMMON RATIO,LAPCON(2),INTENS(3),FRAME(512,512),DIAG
               C*****************************************************************
               C
               C               COMMON BLOCK PARAMETERS
               C
               C       FRAME   = PICTURE ARRAY
               C       INTENS  = INTENSITY TABLE (VALUES AS FOLLOWS):
               C                 1 = FULL INTENSITY PIXEL CODE
               C                 2 = 66% INTENSITY PIXEL CODE
               C                 3 = 33% INTENSITY PIXEL CODE
               C       RATIO   = ASPECT RATIO FOR ANTIALIASING ROUTINE, USED TO
               C                 DETERMINE SHALLOW AND STEEP (NEAR 45 DEG.) LINES
               C                 WHICH USE A LONGER LAP (SEE LAPCON)
               C       LAPCON  = PIXEL LAP CONSTANT TABLE, USED TO SET THE NUMBER
               C                 PIXELS IN THE TRANSITION (LAP) REGION (VALUES AS
               C                 FOLLOWS):
               C                 1 = STANDARD LAP (INTERMEDIATE SLOPE LINES)
               C                 2 = LONG LAP (FOR SHALLOW AND STEEP LINES)
               C
               C*****************************************************************


               C*****************************************************************
               C
               C       COMPUTE DELTA X AND Y AND SETUP OCTANT
               C
               C*****************************************************************
0004            DELX = ENDX - STARTX
0005            DELY = ENDY - STARTY
0006            DELXY = IABS(DELX) - IABS(DELY)
               C  OCTANT 1 OR 2
0007            IF (DELX.GE.0 .AND. DELY.GE.0) THEN
0008                    M2X = 1
0009                    M2Y = 1
               C  OCTANT 1
0010                    IF (DELXY.GE.0) THEN
0011                            DELA = DELX
0012                            DELB = DELY
```

```
0013                                    M1X = 1
0014                                    M1Y = 0
      C  OCTANT 2
0015                            ELSE
0016                                    DELA = DELY
0017                                    DELB = DELX
0018                                    M1X = 0
0019                                    M1Y = 1
0020                            END IF
      C  OCTANT 3 OR 4
0021                  ELSE IF (DELX.LT.0 .AND. DELY.GE.0) THEN
0022                            M2X = -1
0023                            M2Y = 1
      C  OCTANT 4
0024                            IF (DELXY.GE.0) THEN
0025                                    DELA = -DELX
0026                                    DELB = DELY
0027                                    M1X = -1
0028                                    M1Y = 0
      C  OCTANT 3
0029                            ELSE
0030                                    DELA = DELY
0031                                    DELB = -DELX
0032                                    M1X = 0
0033                                    M1Y = 1
0034                            END IF
      C  OCTANT 5 OR 6
0035                  ELSE IF (DELX.LT.0 .AND. DELY.LT.0) THEN
0036                            M2X = -1
0037                            M2Y = -1
      C  OCTANT 5
0038                            IF (DELXY.GE.0) THEN
0039                                    DELA = -DELX
0040                                    DELB = -DELY
0041                                    M1X = -1
0042                                    M1Y = 0
      C  OCTANT 6
0043                            ELSE
0044                                    DELA = -DELY
0045                                    DELB = -DELX
0046                                    M1X = 0
0047                                    M1Y = -1
0048                            END IF
      C  OCTANT 7 OR 8
0049                  ELSE
0050                            M2X = 1
0051                            M2Y = -1
      C  OCTANT 8
0052                            IF (DELXY.GE.0) THEN
0053                                    DELA = DELX
0054                                    DELB = -DELY
0055                                    M1X = 1
0056                                    M1Y = 0
      C  OCTANT 7
0057                            ELSE
0058                                    DELA = -DELY
0059                                    DELB = DELX
```

```
0060                            M1X = 0
0061                            M1Y = -1
0062                    END IF
0063            END IF
      C*****************************************************************
      C
      C       SETUP LINE DRAWING ALGORITHM
      C
      C*****************************************************************
      C  SET UP CONSTANTS
0064            DEL2B = 2*DELB
0065            DEL2AB = 2*(DELB - DELA)
0066            DELTA = DEL2B - DELA
0067            OLDX = STARTX
0068            OLDY = STARTY
0069            FULL = INTENS(1)

C
      C*****************************************************************
      C
      C       DIAGNOSTIC ROUTINE
      C
      C*****************************************************************
0070            IF (DIAG.GT.0) THEN      ! IN DIAGNOSTIC MODE
0071                    WRITE(6,2030)
0072    2030            FORMAT(' DRAWO SUBROUTINE - STANDARD BRESENHAM')
0073                    WRITE(6,2032) DELA,DELB,DELTA
0074    2032            FORMAT(' A=',I4,' B=',I4,' DELTA=',I5)
0075            END IF
      C*****************************************************************


      C*****************************************************************
      C
      C       DRAW THE LINE
      C
C     C*****************************************************************
      C  OUTPUT THE STARTING POINT
0076            FRAME(OLDX,OLDY) = FULL
      C  DRAW THE REMAINDER OF THE LINE
0077    100     IF (DELA.GT.0) THEN              ! DELA = NO. OF POSITIONS IN LINE
0078                    IF (DELTA.LT.0) THEN
      C  M1 MOVE
0079                            OLDX = OLDX + M1X
0080                            OLDY = OLDY + M1Y
0081                            FRAME(OLDX,OLDY) = FULL
0082                            DELTA = DELTA + DEL2B
0083                    ELSE
      C  M2 MOVE
0084                            OLDX = OLDX + M2X
0085                            OLDY = OLDY + M2Y
0086                            FRAME(OLDX,OLDY) = FULL
0087                            DELTA = DELTA + DEL2AB
0088                    END IF
0089                    DELA = DELA - 1          ! DECREMENT POSITION COUNT
0090                    GOTO 100
0091            END IF
```

```
DRAWO

      C
      C     LINE DRAWING COMPLETED
      C
0092        RETURN
0093        END
```

```
0001            SUBROUTINE DRAW1(STARTX, STARTY, ENDX, ENDY)
      C*****************************************************************
      C
      C         LINE PLOTTING ROUTINE USING ANTIALIASING BRESENHAM ALGORITHM
      C         DEVELOPED BY E. J. DUNNING.  THIS ROUTINE USES THE SIMPLE
      C         VERSION OF THE ANTIALIASING ALGORITHM WHICH ONLY CREATES
      C         TRANSITION REGION LAPPING ON LINES WHICH ARE PRIMARILY
      C         AXIAL IN NATURE.  THESE LINES ARE IDENTIFIED BY A NEGATIVE
      C         INITIAL DELTA VALUE.  COMMENTS IN THIS ROUTINE PERTAINING
      C         TO ANTIALIASING STEEP LINES MAY BE IGNORED.  THE APPEARENCE
      C         OF THE STEEP LINES IS ENHANCED, HOWEVER, BY THIS ROUTINE BY
      C         USING 66% INTENSITY PIXEL VALUES IN THE M1 TRANSITIONS.
      C
      C                                   BY E. JACK DUNNING
      C
      C*****************************************************************
      C
      C                   CALL PARAMETERS
      C
      C         STARTX, STARTY = X & Y COORDINATES OF LINE START POINT
      C         ENDX, ENDY     = X & Y CORRDINATES OF LINE END POINT
      C
      C*****************************************************************
0002            IMPLICIT INTEGER*2(A-Z)
0003            COMMON RATIO, LAPCON(2), INTENS(3), FRAME(512, 512), DIAG
      C*****************************************************************
      C
      C                   COMMON BLOCK PARAMETERS
      C
      C         FRAME    = PICTURE ARRAY
      C         INTENS   = INTENSITY TABLE (VALUES AS FOLLOWS):
      C                      1 = FULL INTENSITY PIXEL CODE
      C                      2 = 66% INTENSITY PIXEL CODE
      C                      3 = 33% INTENSITY PIXEL CODE
      C         RATIO    = ASPECT RATIO FOR ANTIALIASING ROUTINE, USED TO
      C                    DETERMINE SHALLOW AND STEEP (NEAR 45 DEG.) LINES
      C                    WHICH USE A LONGER LAP (SEE LAPCON)
      C         LAPCON   = PIXEL LAP CONSTANT TABLE, USED TO SET THE NUMBER
      C                    PIXELS IN THE TRANSITION (LAP) REGION (VALUES AS
      C                    FOLLOWS):
      C                      1 = STANDARD LAP (INTERMEDIATE SLOPE LINES)
      C                      2 = LONG LAP (FOR SHALLOW AND STEEP LINES)
      C
      C*****************************************************************


      C*****************************************************************
      C
      C         COMPUTE DELTA X AND Y AND SETUP OCTANT
      C
      C*****************************************************************
0004            DELX = ENDX - STARTX
0005            DELY = ENDY - STARTY
0006            DELXY = IABS(DELX) - IABS(DELY)
      C  OCTANT 1 OR 2
```

```
0007            IF (DELX.GE.0 .AND. DELY.GE.0) THEN
0008                    M2X = 1
0009                    M2Y = 1
        C   OCTANT 1
0010                    IF (DELXY.GE.0) THEN
0011                            DELA = DELX
0012                            DELB = DELY
0013                            M1X = 1
0014                            M1Y = 0
        C   OCTANT 2
0015                    ELSE
0016                            DELA = DELY
0017                            DELB = DELX
0018                            M1X = 0
0019                            M1Y = 1
0020                    END IF
        C   OCTANT 3 OR 4
0021            ELSE IF (DELX.LT.0 .AND. DELY.GE.0) THEN
0022                    M2X = -1
0023                    M2Y = 1
        C   OCTANT 4
0024                    IF (DELXY.GE.0) THEN
0025                            DELA = -DELX
0026                            DELB = DELY
0027                            M1X = -1
0028                            M1Y = 0
        C   OCTANT 3
0029                    ELSE
0030                            DELA = DELY
0031                            DELB = -DELX
0032                            M1X = 0
0033                            M1Y = 1
0034                    END IF
        C   OCTANT 5 OR 6
0035            ELSE IF (DELX.LT.0 .AND. DELY.LT.0) THEN
0036                    M2X = -1
0037                    M2Y = -1
        C   OCTANT 5
0038                    IF (DELXY.GE.0) THEN
0039                            DELA = -DELX
0040                            DELB = -DELY
0041                            M1X = -1
0042                            M1Y = 0
        C   OCTANT 6
0043                    ELSE
0044                            DELA = -DELY
0045                            DELB = -DELX
0046                            M1X = 0
0047                            M1Y = -1
0048                    END IF
        C   OCTANT 7 OR 8
0049            ELSE
0050                    M2X = 1
0051                    M2Y = -1
        C   OCTANT 8
0052                    IF (DELXY.GE.0) THEN
0053                            DELA = DELX
```

```
0054                            DELB = -DELY
0055                            M1X = 1
0056                            M1Y = 0
      C  OCTANT 7
0057                    ELSE
0058                            DELA = -DELY
0059                            DELB = DELX
0060                            M1X = 0
0061                            M1Y = -1
0062                    END IF
0063            END IF
      C*************************************************************************
      C
      C       SET-UP LINE DRAWING ALGORITHM
      C
      C*************************************************************************
      C  SET-UP CONSTANTS
0064            DEL2B = 2*DELB
0065            DEL2AB = 2*(DELB - DELA)
0066            DELTA = DEL2B - DELA
0067            OLDX = STARTX
0068            OLDY = STARTY
      C  SET-UP ANTIALIASING
0069            FULL = INTENS(1)
0070            IMED = INTENS(2)
0071            IF (DELTA.GE.0) THEN                ! DIAGONAL TYPE LINES
0072                    ANTI1 = -DEL2B              ! DEFAULT VALUE
0073                    IMIN = INTENS(2)            ! SET MIN TO MED INTENSITY
0074                    TYPE = 3
0075            ELSE                                ! AXIAL TYPE LINES
0076                    IF ((RATIO*DELB).LE.DELA) THEN  ! SHALLOW AXIAL
0077                            ANTI1 = -LAPCON(2) * DELB       ! LONG LAP
0078                            TYPE = 1
0079                    ELSE                        ! STANDARD AXIAL
0080                            ANTI1 = -LAPCON(1) * DELB       ! STANDARD LAP
0081                            IF (DELTA.GE.ANTI1) ANTI1 = -DEL2B
                                        ! MINIMUM TRANSITION REGION (LAP) LENGTH
0082                            TYPE = 2
0083                    END IF
0084                    IMIN = INTENS(3)            ! STANDARD MINIMUM INTENSITY
0085                    DELTA = DELTA + ANTI1               ! CORRECT SYMMETRY
0086            END IF
0087            ANTI2 = ANTI1 * 2


      C*************************************************************************
      C
      C       DIAGNOSTIC ROUTINE
      C
      C*************************************************************************
0088            IF (DIAG.GT.0) THEN     ! IN DIAGNOSTIC MODE
0089                    WRITE(6,2030)
0090    2030            FORMAT(' DRAW1 SUBROUTINE - ANTIALIAS ALG. NO. 1')
0091                    WRITE(6,2032) DELA,DELB,DELTA,ANTI1,ANTI2,IMED,IMIN,TYP
0092    2032            FORMAT(' A=',I4,' B=',I4,' DELTA=',I5,' A1=',I5,
           *            ' A2=',I5,' MED=',I4,' MIN=',I4,' TYPE=',I2)
0093            END IF
```

```
      C*************************************************************************
      C*************************************************************************
      C
      C        DRAW THE LINE
      C
      C*************************************************************************
      C  OUTPUT THE STARTING POINT
0094            IF (DELTA.LT.ANTI2) THEN
0095                    FRAME(OLDX,OLDY) = FULL
0096            ELSE
0097                    FRAME(OLDX,OLDY) = IOR(FRAME(OLDX,OLDY),IMED)
0098            END IF
      C  DRAW THE REMAINDER OF THE LINE
0099    100     IF (DELA.GT.0) THEN                ! DELA = NO. OF POSITIONS IN LIN
0100                    IF (DELTA.LT.0) THEN
      C  M1 MOVE
0101                            NEWX = OLDX + M1X
0102                            NEWY = OLDY + M1Y
0103                            IF (DELTA.LT.ANTI2) THEN
0104                                    FRAME(NEWX,NEWY) = FULL
0105                            ELSE IF (DELTA.LT.ANTI1) THEN
0106                                    FRAME(NEWX,NEWY) =
     *                                          IOR(FRAME(NEWX,NEWY),IMED)
0107                                    FRAME(OLDX+M2X,OLDY+M2Y) =
     *                                          IOR(FRAME(OLDX+M2X,OLDY+M2Y),IMI
0108                            ELSE
0109                                    FRAME(NEWX,NEWY) =
     *                                          IOR(FRAME(NEWX,NEWY),IMIN)
0110                                    FRAME(OLDX+M2X,OLDY+M2Y) =
     *                                          IOR(FRAME(OLDX+M2X,OLDY+M2Y),IME
0111                            END IF
0112                            OLDX = NEWX
0113                            OLDY = NEWY
0114                            DELTA = DELTA + DEL2B
0115                    ELSE
      C  M2 MOVE
0116                            OLDX = OLDX + M2X
0117                            OLDY = OLDY + M2Y
0118                            FRAME(OLDX,OLDY) = FULL
0119                            DELTA = DELTA + DEL2AB
0120                    END IF
0121                    DELA = DELA - 1            ! DECREMENT POSITION COUNT
0122                    GOTO 100
0123            END IF
      C
      C        LINE DRAWING COMPLETED
      C
0124            RETURN
0125            END
```

APPENDIX B

```
DEFAULT NANOP CCNOP LDNOP SB ALUZ YD CARO SSO ALUBR MDRIKD MARIKA

ORG 200

;++
;       VECT/LR - IDL LOW RESOLUTION VECT ROUTINE
;
;       VERSION 1.2
;       SCH
;
;***    MODIFIED 26-OCT-81
;       SCH
;       BUG IN SHADE BIT DUPLICATION FIXED
;++

;++
;       COPYRIGHT 1981 IKONAS GRAPHICS SYSTEMS, INC.
;++
```

```
DEFAULT NANOP CCNOP LDNOP SB ALUZ YD CARO SSO ALUBR MDRIKD MARIKA

ORG 6000

;++
;       VECTAA1 - LOW RESOLUTION ANTIALIASING VECT ROUTINE
;
;
;       CALLING FORMAT:
;
;               VECTAA  <VLISTADDR>
;
;       VECTOR LIST FORMAT:
;
;       WORD    UPPER 16 BITS           LOWER 16 BITS
;
;        0          Y-COORD                 X-COORD
;        1       D/M,E,SHADE          Z-COORD (IGNORED)
;       ...        ...                        ...
;
;       (CONTROL/SHADE)\Z COORD. WORD:
;
;       BIT 31          - DRAW=1, MOVE=0
;       BIT 30          - END LIST=1, NO END LIST=0
;       BIT 29          - UNUSED
;       BIT 28          - UNUSED
;       BITS 27-24      - RESERVED
;       BITS 23-16      - VECTOR SHADE
;++
;$INCLUDE IDLSYS.EXP
;++
;       IDLSYS - IDL INTERPRETER/DISPATCHER SYSTEM CONFIGURATION EQUATES
;
;++

;++
;++


SYSFLAG=0               ;ADDRESS OF GENERAL IDL SYSTEM FLAG WORD.
ITABLE=1                ;ADDRESS OF IDL INSTRUCTION RELOC. INDEX TABLE.
ERRWD=ITABLE+256.       ;ADDRESS OF IDL SYSTEM ERROR FLAG WORD.
INDFLAG=ERRWD+1         ;ADDRESS OF THE COPY OF THE CURRENT IDL
                        ;INSTRUCTION INDIRECT ADDRESSING FLAG WORD.
RPNT=INDFLAG+1          ;ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
SYSCOEF=RPNT+1          ;BASE ADDRESS FOR SOFTWARE TRANSFORM MATRICES
INSTR=500               ;ADDRESS OF FIRST IDL INSTRUCTION.
TSTACK=7777             ;ADDRESS OF HIGHEST SCRATCHPAD LOCATION.
BSTACK=TSTACK-500.      ;LOWEST ALLOWABLE STACK LOCATION.

;++
;       IDL RUNTIME ERROR CODE DEFINITIONS.
;++

ERR1=1                  ;IDL STACK UNDERFLOW ERROR CODE
ERR2=2                  ;IDL STACK OVERFLOW ERROR CODE.

;++
;       END OF IDL SYSTEM EQUATES
;++
;$INCLUDE IDLEQS.EXP
```

```
;++
;       IDLEGS - IDL SYSTEM ROUTINE ADDRESS EQUATES
;
;++

GPARMADR=22     ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS)
GPARMVAL=35     ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE)
GPARMSPA=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS SPECIAL)
GPARMSPV=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE SPECIAL)
IDLPUSH=54      ;IDL SYSTEM ROUTINE - PUSH Q ONTO IDL SYSTEM STACK
IDLPOP=70       ;IDL SYSTEM ROUTINE - POP Q OFF OF IDL SYSTEM STACK
IDLEXIT=3       ;IDL SYSTEM ROUTINE - EXIT AND FETCH NEXT IDL INSTRUCTION


;++
;       MISCELLANEOUS PROGRAM EQUATES.
;++

MASK8=377               ;EIGHT BIT SHADE MASK
MASK9=777               ;NINE BIT COORDINATE MASK (SEE NOTE).
MASK10=1777             ;TEN BIT MASK
MASK16=-1               ;SIXTEEN BIT MASK (HALFWORD)
YCONV=1777              ;Y COORD INVERTER VALUE
DELTA=2                 ;COORDINATE DISPLACEMENT VALUE (USED BY DIS1,DIS2)

ZERO= 000 000           ; CONSTANT USED TO CLEAR UPPER DATA REGISTER

;++
;       MAIN PARAMETER FETCHING.
;++

VECTAA: JSBDF GPARMVAL                  ;GET INPUT VERTEX LIST A
        SQ PS BD B1                     ;R1=VERT. PNTR.
                                        ;vertex pointer contains the
                                        ; address in scratchpad RAM of
                                        ; the next vertex word pair

        B1 RLIMM CAR1 SMR BD 1          ;PRE-DECREMENT VERTEX POINTER

;++
;       GET NEXT VERTEX.
;++

;*****************************************************************************
;           LOAD NEW YFIN\XFIN FROM SCRATCHPAD RAM TO REG. 2
;*****************************************************************************
```

```
;       NOTE:
;             THE COORDINATES ARE ACTUALLY MASKED TO 9 BITS. SINCE THE
;       COORDINATES WILL BE MULTIPLIED BY 2, THE END RESULT IS COORDINATES
;       EFFECTIVELY MASKED TO 10 BITS.
;++
                                                ;RETURN MASKED AND SHIFTED
                                                ;YFIN\XFIN TO REG. 2

;*****************************************************************************
;                       EXTRACT YFIN AND XFIN FROM REG. 2
;                   AND PLACE IN REG. 4 AND REG. 5 RESPECTIVELY
;*****************************************************************************

;*****************************************************************************
;                                PREPARE FOR EXIT
;       STORE ENDPOINT AT THE ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
;*****************************************************************************

        B2 PS ALUMDR  LDUDR 202         ;MDR=CURRENT ABS.  YFIN\XFIN
        RIMM PR ALUMAR  RPNT            ;MAR=202\RPNT
EOL:    IKWR JMPDF IDLEXIT              ;WRITE CURRENT POINT, THEN EXIT.

;++
;       SUBROUTINES.
;++


;++
;       VECDIS - GIVEN THE VECTOR END POINTS, VECDIS DETERMINES THE
;                CHANGES (OR DELTAS) IN THE X AND Y  COORDINATES BET-
;                WEEN POINTS OF THE VECTOR.
;
;       ON ENTRY:
;         R4=FINAL Y COORDINATE
```

```
;          R5=FINAL X COORDINATE
;          R6=YFIN\XFIN
;          R7=INITIAL X COORDINATE
;          R8=INITIAL Y COORDINATE
;
;        WORKING VARIABLES:
;          R9=DX
;          R10=DY
;          R11=DISPLACEMENT 1
;          R12=DISPLACEMENT 2
;
;        IDL SYSTEM REGISTERS - DO NOT DESTROY!
;          R14=SOFTWARE STACK POINTER
;          R15=INSTRUCTION POINTER
```

```
;
;       HERE BEGINS THE TOTALLY NEW CODE FOR DRAWING ANTIALIASED LINES
;
;*************************************************************************

;*************************************************************************
;                          REGISTER ALLOCATION
;*************************************************************************

;                                  ALLOCATED FOR:

;REGISTER        ON ENTRY                    IN PROCESS        ON EXIT
;
;     0        2*DELTA(B)                      SAME            SAME
;     1        VERTEX POINTER                  SAME            SAME
;     2        YFIN\XFIN                       SAME            SAME
;     3        CONTROL\SHADE\Z COORD           SAME            SAME
;     4        YFIN                            SAME            SAME
;     5        XFIN                            SAME            SAME
;     6        YINIT\XINIT                     SAME            SAME
;     7        XINIT                           DB              ANTI1
;     8        YINIT                           DB              ANTI2
;     9        DX                              SAME            IMIN
;    10        DY                              DB              DEL2AB = 2*[DB - DA]
;    11        DIS1                            SAME            SAME
;    12        DIS2                            SAME            SAME
;    13        DELTA(A)                        (DA - N)        ZERO
;    14        SOFTWARE STACK PNTR             SAME            SAME
;    15        INSTRUCTION PNTR                SAME            SAME

;*************************************************************************
;
;                       set shade constants (numbers in octal)
;
;    A shade value is specified as a 16-bit value which is used to access
;    a 256 place color lookup table in the RTI system.  Hardware intialization,
;    prior to running this microcode, sets the Ikonas video output channel for
;    pseudocolor red.  In this mode, only the low 8 bits of the data at
;    each pixel location are used to address the color lookup table.  This
;    color lookup table is initialized by the IDL program which calls this
;    microcode.  This initialization amounts to loading the three locations
;    in the lookup table, those addressed by the values this microcode
;    places in the frame buffer, with the appropriate intensity values.
;
;*************************************************************************

FULL=177777     ; hex FFFF - position 255 in lookup table
MED=114631      ; hex 9999 - position 153 in color lookup
MIN=042104      ; hex 4444 - position 68 in color lookup



;*************************************************************************
;               set up constants to be input to Ikonas 16 bit counter
;           for use in keeping track of the number of left shifts used to
;                                implement multiplication
;*************************************************************************
```

```
LAPCON1=177777                  ; X4 MULTIPLIER:   1'S COMPL. OF ZERO
LAPCON2=177775                  ; X16 MULTIPLIER:  1'S COMPL. OF 2
RATIO=177774                    ; X32 MULTIPLIER:  1'S COMPL. OF 3

;*************************************************************************
;                       CLEAR UPPER DATA REGISTER
;*************************************************************************

DRAW:     LDUDR ZERO
;*************************************************************************
;
;                       CALCULATE DEL
;
;*************************************************************************

          RA0 B13 CAR1 RMS QD

;*************************************************************************
;                       DETERMINE IF DEL IS POSITIVE OR NEGATIVE
;*************************************************************************

          CCNEG JMPDF NEGDEL

;*************************************************************************
;               FOR DEL >= 0, SET CONSTANTS FOR DIAGONAL LINE
;*************************************************************************

POSDEL:   RA0 CAR1 MR B7 BD             ; ANTI1 (R7) GETS -2*DELTA(B)
MINHI:    RLIMM PR B9 BD MED            ; PASS MED. INTENSITY VALUE TO MIN REG.
          SQ RA13 CAR1 SMR B10 BD       ; SET UP [(2*DB)-(2*DA)] FOR USE LATER
          JMPDF ANTI2

;*************************************************************************
;                       SET CONSTANTS FOR AXIAL LINES
;*************************************************************************

NEGDEL:   RA0 PR B10 RAFBD              ; D(B) = [2*D(B)]/2
          RA10 PR B8 BD                 ; COPY D(B) INTO REG.8

          LDCNT RATIO                   ; SET UP MULTIPLY CONSTANT IN COUNTER
MULT:     B8 PS LAFBD NCCCNTZ JMPCNT MULT  ; MULT. RATIO * DB

          RA13 B8 CAR1 RMS              ; D(A) - RATIO*D(B)

          RA10 PR B7 BD CCNEG JMPDF STANDRD
                                        ; pass D(B) to reg.7
                                        ; IF RESULT < 0, RATIO*D(B) > D(A)
                                        ;       THEN STANDARD
                                        ; ELSE RES. >= 0,RATIO*D(B) <= D(A)
          ;

;*************************************************************************
;               NOW SET CONSTANTS FOR SHALLOW AXIAL LINES
;*************************************************************************

SHALLOW:  LDCNT LAPCON2                 ; MULTIPLY LAPCON2 * DB
SHMULT:   B7 PS LAFBD NCCCNTZ JMPCNT SHMULT
```

```
          B7 CAR1 MS BD                 ; -LAPCON2 * DB TO REG.7

          JMPDF MINLO


;*************************************************************************
;                  SET CONSTANTS FOR STANDARD AXIAL LINES
;*************************************************************************

STANDRD:  LDCNT LAPCON1                 ; LAPCON1 * DB
STMULT:   B7 PS LAFBD NCCCNTZ JMPCNT STMULT

          B7 CAR1 MS BD                 ; ANTI1 = -LAPCON1 * DB


;*************************************************************************
;              SET MINIMUM TRANSITION REGION LENGTH IF DEL FALLS
;              INTO SECOND HALF OF TRANSITION REGION
;*************************************************************************

          SQ RA7 CAR1 SMR MINLO         ; COMPARE [DEL - ANTI1]
                                        ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                        ; IF RESULT < 0 THEN DEL < ANTI1
          NCCNEG JMPRDF MINREGN         ; ELSE RESULT >= 0 AND DEL >= ANTI1

MINREGN:  RAO CAR1 MR B7 BD             ; SET ANTI1 = - [2*DB]

MINLO:    RLIMM PR B9 BD MIN            ; PASS LOW INTENSITY VALUE TO MIN REG.

;*************************************************************************
;          SET UP THE CONSTANT [(2*DB) - (2*DA)] IN REG. 10 FOR USE LATER
;*************************************************************************

          SQ RA13 CAR1 SMR B10 BD

;*************************************************************************
;                              SYMMETRY CORRECTION
;*************************************************************************

          SQ RA7 RPS QD                 ; DEL = DEL + ANTI1

;*************************************************************************
;                                 SET ANTI2
;*************************************************************************

ANTI2:    RA7 PR LAFBD B8               ; ANTI2 = ANTI1 * 2


;*************************************************************************
;                           OUTPUT THE STARTING POINT
;*************************************************************************

          SQ RA8 CAR1 SMR PT1FULL       ; DEL - ANTI2
                                        ; ANTI2 IS ALWAYS NEG.
                                        ; DEL MAY BE POS. OR NEG. HERE
                                        ; IF DEL POS.
                                        ;    THEN RESULT > 0, DEL > ANTI2

                                        ; IF DEL NEG. THEN
                                        ;    IF RESULT < 0, THEN DEL < ANTI2
          NCCNEG JMPRDF PT1MED          ;    ELSE RESULT >= 0 , DEL >= ANTI2
```

```
PT1FULL:   RLIMM PR ALUMDR FULL          ; LOAD MDR WITH FULL INTENS. VALUE
           B6 PS ALUMAR BD CCMEMAC JMPDF .
OUTPUT1:   LRESWR MASHIKA JMPDF DRAWLN


PT1MED:    RLIMM PR ALUMDR MED           ; LOAD MDR W. MEDIUM INTENS. VALUE
           B6 PS ALUMAR BD CCMEMAC JMPDF .
           LRESWR MASHIKA

;*************************************************************************
;                        DRAW THE REMAINDER OF THE LINE
;*************************************************************************

;*************************************************************************
;                                  EXIT IF DA = 0
;*************************************************************************

DRAWLN:    B13 PS BD

           SQ PS QD CCZERO JMPDF EXIT    ; SET UP FOR DEL COMPARISON
                                         ; EXIT IF D(A) IS ZERO

;*************************************************************************
;                          CHECK FOR M1 MOVE ( DEL < 0 )
;*************************************************************************

           SQ RA8 CAR1 SMR NCCNEG JMPDF M2MOVE   ; SET UP FOR COMP. DEL & ANTI2
                                                 ; DEL .GE. 0 MEANS M2MOVE

;*************************************************************************
;                                  M1 MOVE
;                CHECK TO SEE IF NOT YET IN TRANSITION REGION
;        IF DEL < ANTI2 THEN NOT YET IN TRANSITION REGION
;                          ELSE DEL >= ANTI2, HAVE ENTERED TRANSITION REGION
;*************************************************************************

M1MOVE:    NCCNEG JMPDF M1TREG   ; TEST [DEL - ANTI2]
                                 ; BOTH DEL AND ANTI2 NEG. HERE
                                 ; IF RESULT < 0
                                 ;    THEN DEL < ANTI2
                                 ; ELSE RESULT >= 0
                                 ;    DEL >= ANTI2

;*************************************************************************
;                          WRITE OUT FULL INTENSITY PIXEL
;   LOAD MDR BEFORE ENTERING WAIT LOOP.  THIS IS TO ALLOW SOME DEGREE OF
;   PIPELINING TO TAKE PLACE IN THAT BY LOADING THE MDR BEFORE THE WAIT
;   LOOP INSTEAD OF AFTER IT, YOU ARE GIVING THE MEMORY AN ADDITIONAL
;   200 NSEC. TO BECOME AVAILABLE.
;*************************************************************************

M1FULL:    RLIMM PR ALUMDR FULL
           B6 RA11 CARHO RPS ALUMAR     NCCMEMAC JMPDF OUTPUT2
WAIT2:     RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT2:   LRESWR MASHIKA JMPDF UPDATE

;*************************************************************************
;            IF DEL < ANTI1 THEN IN FIRST HALF OF TRANSITION REGION
;                           ELSE IN SECOND HALF OF TRANSITION REGION
```

```
; ************************************************************************

M1TREG:   SQ RA7 CAR1 SMR                    ; DEL - ANTI1
                                             ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                             ; IF RESULT < 0
                                             ;    THEN DEL < ANTI1
          NCCNEG JMPDF M1MIN                 ; ELSE RESULT >= 0
                                             ;    DEL >= ANTI1


; ************************************************************************
;                    FIRST HALF OF TRANSITION REGION
;              OUTPUT MEDIUM INTENSITY PIXEL IN M1 DIRECTION
; ************************************************************************

M1MED:    B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
OUTPUT3:  LRESWR DFIKD MASHIKA MED

; ************************************************************************
;              OUTPUT MINIMUM INTENSITY PIXEL IN M2 DIRECTION
; ************************************************************************

M2MIN:    B9 PS ALUMDR                       ; PASS ASSIGNED MIN. VALUE TO MDR
          B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
OUTPUT4:  LRESWR MASHIKA JMPDF UPDATE        ;    MAY BE ABLE TO USE THE MIN
                                             ;    DATA FIELD AS ORIGINALLY
                                             ;    WRITTEN SINCE WHEN THIS LOOP IS
                                             ;    USED, B9 = MIN, NOT MED

; ************************************************************************
;    AT THIS POINT, DEL > ANTI1 SO BEYOND MIDWAY POINT IN TRANSITON REGION
;              OUTPUT MINIMUM INTENSIT PIXEL IN M1 DIRECTION
; ************************************************************************

M1MIN:    B9 PS ALUMDR
          B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESWR MASHIKA

; ************************************************************************
;              OUTPUT MEDIUM INTENSITY PIXEL IN M2 DIRECTION
; ************************************************************************

M2MED:    B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESWR DFIKD MASHIKA MED

; ************************************************************************
;           NOW UPDATE POSITION BY ADDING DIS1 TO BOTH COORDINATES
;                AND BY SETTING DEL = DEL + [2 * DB]
; ************************************************************************

UPDATE:   B6 RA11 CARHO RPS BD
          SQ RAO RPS QD JMPDF DECDA          ; DEL = DEL + [2 * DB]
                                             ; GOTO DECREMENT DA (EOL COUNTER)

; ************************************************************************
;                                M2 MOVE
;       UPDATE POSITION, OUTPUT FULL INTENSITY PIXEL, UPDATE DEL
; ************************************************************************

M2MOVE:   B6 RA12 CARHO RPS ALUMAR BD NCCMEMAC JMPDF OUTPUT5 ; (Y\X)=(Y\X)+DIS2
RA6 PR ALUMAR CCMEMAC JMPDF .
```

```
OUTPUT5:   LRESWR DFIKD MASHIKA FULL

;*****************************************************************************
;                          UPDATE DEL FOR M2 MOVE
;        DEL = DEL + [(2*DB) - (2*DA)]
;        THE VALUE ADDED TO DEL HAS ALREADY BEEN COMPUTED AND IS IN  R10
;*****************************************************************************

           SQ RA10 RPS GD                  ; DEL = DEL + [(2*DB) - (2*DA)]


;*****************************************************************************
;                          DECREMENT DA (END OF LINE COUNTER)
;*****************************************************************************


DECDA:     RLIMM B13 CAR1 SMR BD 2         ; DECREMENT DELTA A
                                           ; DECREMENT BY TWO FOR LORES
                                           ; HARDWARE PECULIARITY
           JMPDF DRAWLN                    ; GOTO TOP OF DRAW LOOP

;*****************************************************************************
;                          RETURN TO CALLING PROGRAM
;*****************************************************************************

EXIT:      NARETN


END                                        ; ASSEMBLER DIRECTIVE
```

```
DEFAULT NANOP CCNOP LDNOP SB ALUZ YD CARO SSO ALUBR MDRIKD MARIKA


ORG 7000
;++
;       VECTAA2 - LOW RESOLUTION ANTIALIASING VECT ROUTINE
;
;
;       CALLING FORMAT:
;
;               VECTAA   <VLISTADDR>
;
;       VECTOR LIST FORMAT:
;
;       WORD    UPPER 16 BITS           LOWER 16 BITS
;
;        0          Y-COORD                 X-COORD
;        1       D/M, E, SHADE          Z-COORD (IGNORED)
;       ...        ...                          ...
;
;       (CONTROL/SHADE)\Z COORD. WORD:
;
;       BIT 31          - DRAW=1, MOVE=0
;       BIT 30          - END LIST=1, NO END LIST=0
;       BIT 29          - UNUSED
;       BIT 28          - UNUSED
;       BITS 27-24      - RESERVED
;       BITS 23-16      - VECTOR SHADE
;++
;$INCLUDE IDLSYS.EXP
;++
;       IDLSYS - IDL INTERPRETER/DISPATCHER SYSTEM CONFIGURATION EQUATES
;
;++

;++
;++


SYSFLAG=0               ;ADDRESS OF GENERAL IDL SYSTEM FLAG WORD.
ITABLE=1                ;ADDRESS OF IDL INSTRUCTION RELOC. INDEX TABLE.
ERRWD=ITABLE+256.       ;ADDRESS OF IDL SYSTEM ERROR FLAG WORD.
INDFLAG=ERRWD+1         ;ADDRESS OF THE COPY OF THE CURRENT IDL
                        ;INSTRUCTION INDIRECT ADDRESSING FLAG WORD.
RPNT=INDFLAG+1          ;ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
SYSCOEF=RPNT+1          ;BASE ADDRESS FOR SOFTWARE TRANSFORM MATRICES
INSTR=500               ;ADDRESS OF FIRST IDL INSTRUCTION.
TSTACK=7777             ;ADDRESS OF HIGHEST SCRATCHPAD LOCATION.
BSTACK=TSTACK-500.      ;LOWEST ALLOWABLE STACK LOCATION.

;++
;       IDL RUNTIME ERROR CODE DEFINITIONS.
;++

ERR1=1                  ;IDL STACK UNDERFLOW ERROR CODE
ERR2=2                  ;IDL STACK OVERFLOW ERROR CODE.

;++
;       END OF IDL SYSTEM EQUATES
;++
;$INCLUDE IDLEQS.EXP
```

```
;       IDLEQS - IDL SYSTEM ROUTINE ADDRESS EQUATES
;
;++

GPARMADR=22     ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS)
GPARMVAL=35     ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE)
GPARMSPA=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS SPECIAL)
GPARMSPV=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE SPECIAL)
IDLPUSH=54      ;IDL SYSTEM ROUTINE - PUSH Q ONTO IDL SYSTEM STACK
IDLPOP=70       ;IDL SYSTEM ROUTINE - POP Q OFF OF IDL SYSTEM STACK
IDLEXIT=3       ;IDL SYSTEM ROUTINE - EXIT AND FETCH NEXT IDL INSTRUCTION


;++
;       MISCELLANEOUS PROGRAM EQUATES.
;++

MASK8=377               ;EIGHT BIT SHADE MASK
MASK9=777               ;NINE BIT COORDINATE MASK (SEE NOTE).
MASK10=1777             ;TEN BIT MASK
MASK16=-1               ;SIXTEEN BIT MASK (HALFWORD)
YCONV=1777              ;Y COORD INVERTER VALUE
DELTA=2                 ;COORDINATE DISPLACEMENT VALUE (USED BY DIS1,DIS2)

ZERO=0                  ;CONSTANT USED FOR CLEARING THE UPPER DATA REGISTER


;++
;       MAIN PARAMETER FETCHING.
;       all IDL subroutines pass parameters though the Q register
;++

VECTAA: JSBDF GPARMVAL                          ;GET INPUT VERTEX LIST A
        SQ PS BD B1                             ;R1=VERT. PNTR.
                                                ;vertex pointer contains the
                                                ; address in scatchpad RAM of
                                                ; the next vertex word pair

        B1 RLIMM CAR1 SMR BD 1                  ;PRE-DECREMENT VERTEX POINTER

;++
;       GET NEXT VERTEX.
;++

                                        ;PASS PREVIOUS YFIN TO NEW YINIT
                                        ;PASS PREVIOUS XFIN TO NEW XINIT
                                        ;PASS PREVIOUS YFIN\XFIN
                                        ; TO NEW YINIT\XINIT

;*****************************************************************************
;           LOAD NEW YFIN\XFIN FROM SCRATCHPAD RAM TO REG.2
;*****************************************************************************

                                        ;POINT TO NEXT VERTEX
                                        ;READ NEXT VERTEX Y\X

                                        ;READ NEXT S\Z, LOAD MASK
;++
```

```
;       LOW RESOLUTION COORDINATE TRANSLATION (HARDWARE PECULIARITY).
;       NOTE:
;             THE COORDINATES ARE ACTUALLY MASKED TO 9 BITS. SINCE THE
;       COORDINATES WILL BE MULTIPLIED BY 2, THE END RESULT IS COORDINATES
;       EFFECTIVELY MASKED TO 10 BITS.
; ++
                                                ;RETURN MASKED AND SHIFTED
                                                ;YFIN\XFIN TO REG. 2

;*****************************************************************************
;                    EXTRACT YFIN AND XFIN FROM REG. 2
;                AND PLACE IN REG. 4 AND REG. 5 RESPECTIVELY
;*****************************************************************************
```



```
;*****************************************************************************
;                              PREPARE FOR EXIT
;       STORE ENDPOINT AT THE ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
;*****************************************************************************

        B2 PS ALUMDR  LDUDR 202         ;MDR=CURRENT ABS. YFIN\XFIN
        RIMM PR ALUMAR  RPNT            ;MAR=202\RPNT
EOL:    IKWR JMPDF IDLEXIT              ;WRITE CURRENT POINT, THEN EXIT.

; ++
;       SUBROUTINES.
; ++


; ++
;       VECDIS - GIVEN THE VECTOR END POINTS, VECDIS DETERMINES THE
;                CHANGES (OR DELTAS) IN THE X AND Y  COORDINATES BET-
;                WEEN POINTS OF THE VECTOR.
;
```

```
;       ON ENTRY:
;         R4=FINAL Y COORDINATE
;         R5=FINAL X COORDINATE
;         R6=YFIN\XFIN
;         R7=INITIAL X COORDINATE
;         R8=INITIAL Y COORDINATE
;
;       WORKING VARIABLES:
;         R9=DX
;         R10=DY
;         R11=DISPLACEMENT 1
;         R12=DISPLACEMENT 2
;
;       IDL SYSTEM REGISTERS - DO NOT DESTROY!
;         R14=SOFTWARE STACK POINTER
;         R15=INSTRUCTION POINTER
```

C 2

```
;*****************************************************************************
;
;       HERE BEGINS THE TOTALLY NEW CODE FOR DRAWING ANTIALIASED LINES
;
;*****************************************************************************


;*****************************************************************************
;                          REGISTER ALLOCATION
;*****************************************************************************

;                                   ALLOCATED FOR:

;REGISTER        ON ENTRY                    IN PROCESS          ON EXIT
;
;     0        2*DELTA(B)                      SAME              SAME
;     1        VERTEX POINTER                  SAME              SAME
;     2        YFIN\XFIN                       SAME              SAME
;     3        CONTROL\SHADE\Z COORD         C\S\IMIN            SAME
;     4        YFIN                            SAME              SAME
;     5        XFIN                            SAME              SAME
;     6        YINIT\XINIT                     SAME              SAME
;     7        XINIT                             DB              ANTI1
;     8        YINIT                             DB              ANTI2
;     9        DX                      FRAME BUFFER INPUT        SAME
;    10        DY                                DB              DEL2AB = 2*[DB - DA]
;    11        DIS1                            SAME              SAME
;    12        DIS2                            SAME              SAME
;    13        DELTA(A)                      (DA - N)            ZERO
;    14        SOFTWARE STACK PNTR             SAME              SAME
;    15        INSTRUCTION PNTR                SAME              SAME


;*****************************************************************************
;
;                     set shade constants (numbers in octal)
;
;    A shade value is specified as a 4-bit value which is used to access
;    a 16 place color lookup table in the RTI system.  This 4-bit value is
;    repeated 4 times because of another RTI system peculiarity.  The write
;    mask in the RTI system determines which of the four 4-bit fields is used
;    depending on which frame buffer of which display is currently being
;    updated.  Thus, all four shade fields are provided here, other IDL
;    code insures that a shade field gets to the proper frame buffer.
;
;*****************************************************************************

FULL=177777       ; hex FFFF - position 15 in lookup table specified 4 times
MED=114631        ; hex 9999 - position 10 in color lookup table spec'd 4 times
MIN=042104        ; hex 4444 - position 5 in color lookup table spec'd 4 times


;*****************************************************************************
;               set up constants to be input to Ikonas 16 bit counter
;            for use in keeping track of the number of left shifts used to
;                                implement multiplication
;*****************************************************************************

LAPCON1=177777                    ; X4 MULTIPLIER:   1'S COMPL. OF ZERO
```

```
DR CONE=177776                  ; X16 MULTIPLIER: 1'S COMPL. OF 2
RATIO=177774                    ; X32 MULTIPLIER: 1'S COMPL. OF 3

;*****************************************************************************
;                        CLEAR UPPER DATA REGISTER
;       TO PERMIT THE USE OF IMMEDIATE DATA WITH THE DFIKD SPECIFICATION.
;                          ALSO CLEAR THE Z FIELD
;*****************************************************************************

DRAW:     LDUDR MASK16                  ; CAN ALMOST CERTAINLY BE
                                        ; BE PIPELINED
          B2 RIMM RAS BD ZERO           ; CLEAR THE Z FIELD IN REG. 2

          LDUDR ZERO                    ; CAN PROB. PIPELINE
;*****************************************************************************
;
;                         CALCULATE DEL
;
;*****************************************************************************

          RAO B13 CAR1 RMS QD

;*****************************************************************************
;                        DETERMINE IF DEL IS POSITIVE OR NEGATIVE
;*****************************************************************************

          CCNEG JMPDF NEGDEL

;*****************************************************************************
;                FOR DEL >= 0, SET CONSTANTS FOR DIAGONAL LINE
;*****************************************************************************

POSDEL:   RAO CAR1 MR B7 BD             ; ANTI1 (R7) GETS -2*DELTA(B)
MINHI:    B2 RLIMM RORS BD MSD          ; PUT IMIN IN Z FIELD OF REG. 2
          SQ RA13 CAR1 SMR B10 BD       ; SET [(2*DB)-(2*DA)] FOR USE LATER
          JMPDF ANTI2

;*****************************************************************************
;                        SET CONSTANTS FOR AXIAL LINES
;*****************************************************************************

NEGDEL:   RAO PR B10 RAFBD              ; D(B) = [2*D(B)]/2
          RA10 PR B8 BD                 ; COPY D(B) INTO REG. 8

          LDCNT RATIO                   ; SET UP MULTIPLY CONSTANT IN COUNTER
MULT:     B8 PS LAFBD NCCCNTZ JMPCNT MULT  ; MULT. RATIO * DB

          RA13 B8 CAR1 RMS              ; D(A) - RATIO*D(B)
          RA10 PR B7 BD CCNEG JMPDF STANDRD

                                        ; PASS D(B) TO REG. 7
                                        ; IF RESULT < 0, RATIO*D(B) > D(A), STANDARD
                                        ;           ELSE RATIO*D(B) <= D(A), SHALLOW

;*****************************************************************************
;                NOW SET CONSTANTS FOR SHALLOW AXIAL LINES
;*****************************************************************************
```

```
SHALLOW:   LDCNT LAPCON2                    ; MULTIPLY LAPCON2 * DB
SHMULT:    B7 PS LAFBD NCCCNTZ JMPCNT SHMULT

           B7 CAR1 MS BD                    ; -LAPCON2 * DB TO REG. 7

           JMPDF MINLO


; ***************************************************************************
;                    SET CONSTANTS FOR STANDARD AXIAL LINES
; ***************************************************************************

STANDRD:   LDCNT LAPCON1                    ; LAPCON1 * DB
STMULT:    B7 PS LAFBD NCCCNTZ JMPCNT STMULT

           B7 CAR1 MS BD                    ; ANTI1 = -LAPCON1 * DB


; ***************************************************************************
;                SET MINIMUM TRANSITION REGION LENGTH IF DEL FALLS
;                INTO SECOND HALF OF TRANSITION REGION
; ***************************************************************************

           SQ RA7 CAR1 SMR MINLO            ; COMPARE [DEL - ANTI1]
                                            ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                            ; IF RESULT < 0 THEN DEL < ANTI1
           NCCNEG JMPRDF MINREGN            ; ELSE RESULT >= 0 AND DEL >= ANTI1

MINREGN:   RA0 CAR1 MR B7 BD                ; SET ANTI1 = - [2*DB]

MINLO:     B2 RLIMM RORS BD MIN             ; PUT IMIN IN Z FIELD OF REG. 2

; ***************************************************************************
;          SET UP THE CONSTANT [(2*DB) - (2*DA)] IN REG. 10 FOR USE LATER
; ***************************************************************************

           SQ RA13 CAR1 SMR B10 BD

; ***************************************************************************
;                                SYMMETRY CORRECTION
; ***************************************************************************

           SQ RA7 RPS QD                    ; DEL = DEL + ANTI1

; ***************************************************************************
;                                SET ANTI2
; ***************************************************************************

ANTI2:     RA7 PR LAFBD B8                  ; ANTI2 = ANTI1 * 2


; ***************************************************************************
;                            OUTPUT THE STARTING POINT
; ***************************************************************************

           SQ RAB CAR1 SMR PT1FULL          ; DEL - ANTI2
                                            ; ANTI2 IS ALWAYS NEGATIVE
                                            ; DEL MAYY BE POS. OR NEG. HERE
                                            ; IF DEL POS
                                            ;    THEN RESULT > 0, DEL > ANTI2
```

```
                                        ;  IF DEL NEG.
                                        ;     IF RESULT < 0, THEN DEL < ANTI2
          NCCNEG JMPRDF PT1MED          ;     ELSE RESULT >= 0, DEL >= ANTI2

PT1FULL:  RLIMM PR ALUMDR FULL          ; LOAD MDR WITH FULL INTENS. VALUE
          B6 PS ALUMAR BD CCMEMAC JMPDF .
          LRESWR MASHIKA JMPDF DRAWLN

PT1MED:   B6 PS ALUMAR BD CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                    ; OR the pixel value at the addressed
          B9 RLIMM RORS ALUMDR MED      ; location with the MED value to be
          LRESWR MASHIKA                ; output

;*************************************************************************
;                        DRAW THE REMAINDER OF THE LINE
;*************************************************************************

;*************************************************************************
;                              EXIT IF DA = 0
;*************************************************************************

DRAWLN:   B13 PS BD

          SQ PS QD CCZERO JMPDF EXIT    ; SET UP FOR DEL COMPARISON

;*************************************************************************
;                        CHECK FOR M1 MOVE ( DEL < 0 )
;*************************************************************************

          SQ RA8 CAR1 SMR NCCNEG JMPDF M2MOVE   ; SET UP FOR COMP. DEL & ANTI2
                                                ; DEL .GE. 0 MEANS M2MOVE

;*************************************************************************
;                                M1 MOVE
;   CHECK TO SEE IF NOT YET IN TRANSITION REGION
;   IF DEL < ANTI2 THEN NOT YET IN TRANSITION REGION
;                  ELSE DEL >= ANTI2, HAVE ENTERED TRANSITION REGION
;*************************************************************************

M1MOVE:   NCCNEG JMPDF M1TREG           ; TEST [DEL - ANTI2]
                                        ; BOTH DEL AND ANTI2 NEG. HERE
                                        ; IF RESULT < 0
                                        ;    THEN DEL < ANTI2
                                        ; ELSE RESULT >= 0
                                        ;    DEL >= ANTI2

;*************************************************************************
;
;                        WRITE OUT FULL INTENSITY PIXEL
;   LOAD MDR BEFORE ENTERING WAIT LOOP.  THIS IS TO ALLOW SOME DEGREE OF
;   PIPELINING TO TAKE PLACE IN THAT BY LOADING THE MDR BEFORE THE WAIT
;   LOOP INSTEAD OF AFTER IT, YOU ARE GIVING THE MEMORY AN ADDITIONAL
;   200 NSEC. TO BECOME AVAILABLE.
;
;*************************************************************************

M1FULL:   RLIMM PR ALUMDR FULL
```

```
        B6 RA11 CARHO RPS ALUMAR    NCCMEMAC JMPDF OUTPUT2
        RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT2:  LRESWR MASHIKA JMPDF UPDATE

; ***************************************************************************
;           IF DEL < ANTI1 THEN IN FIRST HALF OF TRANSITION REGION
;                          ELSE IN SECOND HALF OF TRANSITION REGION
; ***************************************************************************

M1TREG:   SG RA7 CAR1 SMR                   ; DEL - ANTI1
                                            ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                            ; IF RESULT < 0
                                            ;    THEN DEL < ANTI1
          NCCNEG JMPDF M1MIN                ; ELSE RESULT >= 0
                                            ;    DEL >= ANTI1

; ***************************************************************************
;                    FIRST HALF OF TRANSITION REGION
;              OUTPUT MEDIUM INTENSITY PIXEL IN M1 DIRECTION
; ***************************************************************************

M1MED:    B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                        ; OR pixel value at addressed location
          B9 RLIMM RORS ALUMDR MED          ; with MED value to be output
          LRESWR MASHIKA

; ***************************************************************************
;              OUTPUT MINIMUM INTENSITY PIXEL IN M2 DIRECTION
; ***************************************************************************

M2MIN:    B2 RLIMM RAS ALUMDR MASK16        ; LOAD MDR PRIOR TO WAIT LOOP FOR
                                            ; PIPELINE EFFECT

          B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA                    ;    MAY BE ABLE TO USE THE MIN
          IKBR BD B9                        ;    DATA FIELD AS ORIGINALLY
          B9 RMDR RORS ALUMDR               ;    WRITTEN SINCE WHEN THIS LOOP IS
          LRESWR MASHIKA JMPDF UPDATE       ;    USED, B9 = MIN, NOT MED ??
                                            ;    NO REASON TO CHANGE THIS HOWEVER


; ***************************************************************************
;    AT THIS POINT, DEL > ANTI1 SO BEYOND MIDWAY POINT IN TRANSITON REGION
;              OUTPUT MINIMUM INTENSIT PIXEL IN M1 DIRECTION
; ***************************************************************************

M1MIN:    B2 RLIMM RAS ALUMDR MASK16        ; LOAD MDR PRIOR TO WAIT LOOP
                                            ; FOR PIPELINE EFFECT

          B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                        ; OR pixel value at addressed location
          B9 RMDR RORS ALUMDR               ; with IMIN value to be output
          LRESWR MASHIKA                          ; there may be a wait here
                                                  ; but no wait loop need be
                                                  ; explicitly specified

; ***************************************************************************
;              OUTPUT MEDIUM INTENSITY PIXEL IN M2 DIRECTION
```

```
; ************************************************************************

M2MED:      B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
            LRESRD MASHIKA
            IKBR BD B9                      ; OR pixel value at addressed location
            B9 RLIMM RORS ALUMDR MED        ; with MED value to be output
            LRESWR MASHIKA

; ************************************************************************
;           NOW UPDATE POSITION BY ADDING DIS1 TO BOTH COORDINATES
;                AND BY SETTING DEL = DEL + [2 * DB]
; ************************************************************************

UPDATE:     B6 RA11 CARHO RPS BD            ; [Y\X] + M1 DISPLACEMENT (DIS1)
            SQ RA0 RPS QD JMPDF DECDA       ; DEL = DEL + [2 * DB]
                                            ; GOTO DECREMENT DA (EOL COUNTER)

; ************************************************************************
;                                 M2 MOVE
;        UPDATE POSITION, OUTPUT FULL INTENSITY PIXEL, UPDATE DEL
; ************************************************************************

M2MOVE:     B6 RA12 CARHO RPS ALUMAR BD NCCMEMAC JMPDF OUTPUT5 ; (Y\X)=(Y\X)+DIS2
            RA6 PR ALUMAR JMPDF .
OUTPUT5:    LRESWR DFIKD MASHIKA FULL

; ************************************************************************
;                          UPDATE DEL FOR M2 MOVE
;        DEL = DEL + [(2*DB) - (2*DA)]
;        THE VALUE ADDED TO DEL HAS ALREADY BEEN COMPUTED AND IS IN  R10
; ************************************************************************

            SQ RA10 RPS QD                  ; DEL = DEL + [(2*DB) - (2*DA)]
                                            ; can pipeline w. OUTPUT5

; ************************************************************************
;                          DECREMENT DA (END OF LINE COUNTER)
; ************************************************************************


DECDA:      RLIMM B13 CAR1 SMR BD 2         ; DECREMENT DELTA A
                                            ; DECREMENT BY TWO FOR LORES
                                            ; HARDWARE PECULIARITY
            JMPDF DRAWLN

; ************************************************************************
;                             RETURN TO CALLING PROGRAM
; ************************************************************************

EXIT:       NARETN

END                                         ; ASSEMBLER DIRECTIVE
```

```
DEFAULT NANOP CONOP LONOP SB ALOZ YD CARO SSO ALOBR MORIKO MARIKA

ORG 6000

;++
;       VECTAA1F    - LOW RESOLUTION ANTIALIASING VECT ROUTINE
;
;
;       CALLING FORMAT:
;
;               VECTAA    <VLISTADDR>
;
;       VECTOR LIST FORMAT:
;
;       WORD     UPPER 16 BITS            LOWER 16 BITS
;
;        0          Y-COORD                  X-COORD
;        1        D/M,E,SHADE          Z-COORD (IGNORED)
;       ...         ...                        ...
;
;       (CONTROL/SHADE)\Z COORD.  WORD:
;
;       BIT 31          - DRAW=1, MOVE=0
;       BIT 30          - END LIST=1, NO END LIST=0
;       BIT 29          - UNUSED
;       BIT 28          - UNUSED
;       BITS 27-24      - RESERVED
;       BITS 23-16      - VECTOR SHADE
;++
;$INCLUDE IDLSYS.EXP
;++
;       IDLSYS - IDL INTERPRETER/DISPATCHER SYSTEM CONFIGURATION EQUATES
;
;++

;++
;++


SYSFLAG=0               ;ADDRESS OF GENERAL IDL SYSTEM FLAG WORD.
ITABLE=1                ;ADDRESS OF IDL INSTRUCTION RELOC.  INDEX TABLE.
ERRWD=ITABLE+256.       ;ADDRESS OF IDL SYSTEM ERROR FLAG WORD.
INDFLAG=ERRWD+1         ;ADDRESS OF THE COPY OF THE CURRENT IDL
                        ;INSTRUCTION INDIRECT ADDRESSING FLAG WORD.
RPNT=INDFLAG+1          ;ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
SYSCOEF=RPNT+1          ;BASE ADDRESS FOR SOFTWARE TRANSFORM MATRICES
INSTR=500               ;ADDRESS OF FIRST IDL INSTRUCTION.
TSTACK=7777             ;ADDRESS OF HIGHEST SCRATCHPAD LOCATION.
BSTACK=TSTACK-500.      ;LOWEST ALLOWABLE STACK LOCATION.

;++
;       IDL RUNTIME ERROR CODE DEFINITIONS.
;++

ERR1=1                  ;IDL STACK UNDERFLOW ERROR CODE
ERR2=2                  ;IDL STACK OVERFLOW ERROR CODE.

;++
;       END OF IDL SYSTEM EQUATES
;++
;$INCLUDE IDLEQS.EXP
```

```
;++
;       IDLEGS - IDL SYSTEM ROUTINE ADDRESS EQUATES
;
;++

GPARMADR=22     ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS)
GPARMVAL=35     ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE)
GPARMSPA=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS SPECIAL)
GPARMSPV=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE SPECIAL)
IDLPUSH=54      ;IDL SYSTEM ROUTINE - PUSH Q ONTO IDL SYSTEM STACK
IDLPOP=70       ;IDL SYSTEM ROUTINE - POP Q OFF OF IDL SYSTEM STACK
IDLEXIT=3       ;IDL SYSTEM ROUTINE - EXIT AND FETCH NEXT IDL INSTRUCTION


;++
;       MISCELLANEOUS PROGRAM EQUATES.
;++

MASK8=377               ;EIGHT BIT SHADE MASK
MASK9=777               ;NINE BIT COORDINATE MASK (SEE NOTE).
MASK10=1777             ;TEN BIT MASK
MASK16=-1               ;SIXTEEN BIT MASK (HALFWORD)
YCONV=1777              ;Y COORD INVERTER VALUE
DELTA=2                 ;COORDINATE DISPLACEMENT VALUE (USED BY DIS1,DIS2)

ZERO= 000 000           ; CONSTANT USED TO CLEAR UPPER DATA REGISTER

;++
;       MAIN PARAMETER FETCHING.
;++

VECTAA: JSBDF GPARMVAL                  ;GET INPUT VERTEX LIST A
        SQ PS BD B1                     ;R1=VERT: PNTR.
                                        ;vertex pointer contains the
                                        ; address in scratchpad RAM of
                                        ; the next vertex word pair

        B1 RLIMM CAR1 SMR BD 1          ;PRE-DECREMENT VERTEX POINTER

;++
;       GET NEXT VERTEX.
;++

                                        ;PASS PREVIOUS YFIN TO NEW YINIT
                                        ;PASS PREVIOUS XFIN TO NEW XINIT
                                        ;PASS PREVIOUS YFIN\XFIN
                                        ; TO NEW YINIT\XINIT

;*****************************************************************************
;           LOAD NEW YFIN\XFIN FROM SCRATCHPAD RAM TO REG.2
;*****************************************************************************

                                        ;POINT TO NEXT VERTEX
                                        ;READ NEXT VERTEX Y\X

                                        ;READ NEXT S\Z, LOAD MASK
;++
;       MASK Y\X COORDINATES TO 10 BITS, THEN MULTIPLY BOTH BY 2 FOR
;       LOW RESOLUTION COORDINATE TRANSLATION (HARDWARE PECULIARITY).
```

```
;       NOTE:
;            THE COORDINATES ARE ACTUALLY MASKED TO 9 BITS. SINCE THE
;       COORDINATES WILL BE MULTIPLIED BY 2, THE END RESULT IS COORDINATES
;       EFFECTIVELY MASKED TO 10 BITS.
;++
                                                ;RETURN MASKED AND SHIFTED
                                                ;YFIN\XFIN TO REG. 2

;*****************************************************************************
;                   EXTRACT YFIN AND XFIN FROM REG. 2
;                AND PLACE IN REG. 4 AND REG. 5 RESPECTIVELY
;*****************************************************************************

;*****************************************************************************
;                              PREPARE FOR EXIT
;      STORE ENDPOINT AT THE ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
;*****************************************************************************

        B2 PS ALUMDR  LDUDR 202         ;MDR=CURRENT ABS. YFIN\XFIN
        RIMM PR ALUMAR  RPNT            ;MAR=202\RPNT
EOL:    IKWR JMPDF IDLEXIT              ;WRITE CURRENT POINT, THEN EXIT.

;++
;       SUBROUTINES.
;++


;++
;       VECDIS - GIVEN THE VECTOR END POINTS, VECDIS DETERMINES THE
;                CHANGES (OR DELTAS) IN THE X AND Y  COORDINATES BET-
;                WEEN POINTS OF THE VECTOR.
;
;       ON ENTRY:
;         R4=FINAL Y COORDINATE
```

```
;       R5=FINAL X COORDINATE
;       R6=YFIN\XFIN
;       R7=INITIAL X COORDINATE
;       R8=INITIAL Y COORDINATE
;
;     WORKING VARIABLES:
;       R9=DX
;       R10=DY
;       R11=DISPLACEMENT 1
;       R12=DISPLACEMENT 2
;
;     IDL SYSTEM REGISTERS - DO NOT DESTROY!
;       R14=SOFTWARE STACK POINTER
;       R15=INSTRUCTION POINTER
```

```
;
;       HERE BEGINS THE TOTALLY NEW CODE FOR DRAWING ANTIALIASED LINES
;
;*****************************************************************************


;*****************************************************************************
;                          REGISTER ALLOCATION
;*****************************************************************************

;                                ALLOCATED FOR:

;REGISTER        ON ENTRY                    IN PROCESS        ON EXIT
;
;     0        2*DELTA(B)                      SAME            SAME
;     1        VERTEX POINTER                  SAME            SAME
;     2        YFIN\XFIN                       SAME            SAME
;     3        CONTROL\SHADE\Z COORD           SAME            SAME
;     4        YFIN                            SAME            SAME
;     5        XFIN                            SAME            SAME
;     6        YINIT\XINIT                     SAME            SAME
;     7        XINIT                           DB              ANTI1
;     8        YINIT                           DB              ANTI2
;     9        DX                              SAME            IMIN
;    10        DY                              DB              DEL2AB = 2*[DB - DA]
;    11        DIS1                            SAME            SAME
;    12        DIS2                            SAME            SAME
;    13        DELTA(A)                        (DA - N)        ZERO
;    14        SOFTWARE STACK PNTR             SAME            SAME
;    15        INSTRUCTION PNTR                SAME            SAME


;*****************************************************************************
;
;                       set shade constants (numbers in octal)
;
;    A shade value is specified as a 16-bit value which is used to access
;    a 256 place color lookup table in the RTI system.  Hardware intialization,
;    prior to running this microcode, sets the Ikonas video output channel for
;    pseudocolor red.  In this mode, only the low 8 bits of the data at
;    each pixel location are used to address the color lookup table.  This
;    color lookup table is initialized by the IDL program which calls this
;    microcode.  This initialization amounts to loading the three locations
;    in the lookup table, those addressed by the values this microcode
;    places in the frame buffer, with the appropriate intensity values.
;
;*****************************************************************************

FULL=177777     ; hex FFFF - position 255 in lookup table
MED=114631      ; hex 9999 - position 153 in color lookup
MIN=042104      ; hex 4444 - position 68 in color lookup


;*****************************************************************************
;                set up constants to be input to Ikonas 16 bit counter
;             for use in keeping track of the number of left shifts used to
;                                implement multiplication
;*****************************************************************************

LAPCON1=177777                  ; X4 MULTIPLIER:  1'S COMPL. OF ZERO
```

```
LAPCON2=177775          ; X16 MULTIPLIER: 1'S COMPL. OF 2
RATIO=177774            ; X32 MULTIPLIER: 1'S COMPL. OF 3

;*************************************************************************
;
;               Calculate DEL and Clear Upper Data Register
;
;*************************************************************************

DRAW:     RA0 B13 CAR1 RMS QD LDUDR ZERO

;*************************************************************************
;                       DETERMINE IF DEL IS POSITIVE OR NEGATIVE
;*************************************************************************

          CCNEG JMPDF NEGDEL

;*************************************************************************
;               FOR DEL >= 0, SET CONSTANTS FOR DIAGONAL LINE
;*************************************************************************

POSDEL:   RA0 CAR1 MR B7 BD               ; ANTI1 (R7) GETS -2*DELTA(B)
MINHI:    RLIMM PR B9 BD MED              ; PASS MED. INTENSITY VALUE TO MIN REG.
          SQ RA13 CAR1 SMR B10 BD JMPDF ANTI2    ; [(2*DB)-(2*DA)] FOR USE LATER

;*************************************************************************
;                       SET CONSTANTS FOR AXIAL LINES
;*************************************************************************

NEGDEL:   RA0 PR B10 RAFBD                ; D(B) = [2*D(B)]/2
          RA10 PR B8 BD                   ; COPY D(B) INTO REG.8

          LDCNT RATIO                     ; SET UP MULTIPLY CONSTANT IN COUNTER
MULT:     B8 PS LAFBD NCCCNTZ JMPCNT MULT  ; MULT. RATIO * DB

          RA13 B8 CAR1 RMS                ; D(A) - RATIO*D(B)

          RA10 PR B7 BD CCNEG JMPDF STANDRD
                                          ; pass D(B) to reg.7
                                          ; IF RESULT < 0, RATIO*D(B) > D(A)
                                          ;       THEN STANDARD
                                          ; ELSE RES. >= 0, RATIO*D(B) <= D(A)


;*************************************************************************
;               NOW SET CONSTANTS FOR SHALLOW AXIAL LINES
;*************************************************************************

SHALLOW:  LDCNT LAPCON2                   ; MULTIPLY LAPCON2 * DB
SHMULT:   B7 PS LAFBD NCCCNTZ JMPCNT SHMULT

          B7 CAR1 MS BD JMPDF MINLO                ; -LAPCON2 * DB TO REG.7


;*************************************************************************
;                   SET CONSTANTS FOR STANDARD AXIAL LINES
;*************************************************************************
```

```
STMULT:   B7 PS LAFBD NCCCNTZ JMPCNT STMULT

          B7 CAR1 MS BD                    ; ANTI1 = -LAPCON1 * DB


; ***************************************************************************
;              SET MINIMUM TRANSITION REGION LENGTH IF DEL FALLS
;              INTO SECOND HALF OF TRANSITION REGION
; ***************************************************************************

          SQ RA7 CAR1 SMR MINLO            ; COMPARE [DEL - ANTI1]
                                           ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                           ; IF RESULT < 0 THEN DEL < ANTI1
          NCCNEG JMPRDF MINREGN            ; ELSE RESULT >= 0 AND DEL >= ANTI1

MINREGN:  RA0 CAR1 MR B7 B7                ; SET ANTI1 = - [2*DB]

MINLO:    RLIMM PR B9 BD MIN               ; PASS LOW INTENSITY VALUE TO MIN REG.

; ***************************************************************************
;          SET UP THE CONSTANT [(2*DB) - (2*DA)] IN REG. 10 FOR USE LATER
; ***************************************************************************

          SQ RA13 CAR1 SMR B10 BD

; ***************************************************************************
;                              SYMMETRY CORRECTION
; ***************************************************************************

          SQ RA7 RPS QD                    ; DEL = DEL + ANTI1

; ***************************************************************************
;                                  SET ANTI2
; ***************************************************************************

ANTI2:    RA7 PR LAFBD B8                  ; ANTI2 = ANTI1 * 2


; ***************************************************************************
;                           OUTPUT THE STARTING POINT
; ***************************************************************************

          SQ RA8 CAR1 SMR PT1FULL          ; DEL - ANTI2
                                           ; ANTI2 IS ALWAYS NEG.
                                           ; DEL MAY BE POS. OR NEG. HERE
                                           ; IF DEL POS.
                                           ;    THEN RESULT > 0, DEL > ANTI2

                                           ; IF DEL NEG. THEN
                                           ;    IF RESULT < 0, THEN DEL < ANTI2
          NCCNEG JMPRDF PT1MED             ;    ELSE RESULT >= 0 ,, DEL >= ANTI2

PT1FULL:  RLIMM PR ALUMDR FULL             ; LOAD MDR WITH FULL INTENS. VALUE
          B6 PS ALUMAR BD CCMEMAC JMPDF .

OUTPUT1:  LRESWR MASHIKA B13 PS BD JMPDF DRAWLN  ; SET UP FOR INITIAL
                                                 ; DELTA A (EOL) CHECK

PT1MED:   RLIMM PR ALUMDR MED              ; LOAD MDR W. MEDIUM INTENS. VALUE
          B6 PS ALUMAR BD CCMEMAC JMPDF .
```

```
        LRESWR MASHIKA B13 PS BD        ; SET UP FOR EOL CHECK

;*****************************************************************************
;                       DRAW THE REMAINDER OF THE LINE
;*****************************************************************************

;*****************************************************************************
;                              EXIT IF DA = 0
;*****************************************************************************

                                        ; EXIT IF D(A) IS ZERO

DRAWLN:   SQ PS QD CCZERO NARETN        ; SET UP FOR DEL COMPARISON
;*****************************************************************************
;                       CHECK FOR M1 MOVE ( DEL < 0 )
;*****************************************************************************

          SQ RA8 CAR1 SMR NCCNEG JMPDF M2MOVE   ; SET UP FOR COMP. DEL & ANTI2
                                                ; DEL .GE. 0 MEANS M2MOVE

;*****************************************************************************
;                                  M1 MOVE
;                CHECK TO SEE IF NOT YET IN TRANSITION REGION
;        IF DEL < ANTI2 THEN NOT YET IN TRANSITION REGION
;                        ELSE DEL >= ANTI2, HAVE ENTERED TRANSITION REGION
;*****************************************************************************

                                        ; BOTH DEL AND ANTI2 NEG. HERE
M1MOVE:   SQ RA7 CAR1 SMR NCCNEG JMPDF M1TREG  ; TEST [DEL-ANTI2], DEL-ANTI1
                                        ; IF RESULT < 0
                                        ;    THEN DEL < ANTI2
                                        ; ELSE RESULT >= 0
                                        ;    DEL >= ANTI2

;*****************************************************************************
;                       WRITE OUT FULL INTENSITY PIXEL
;    LOAD MDR BEFORE ENTERING WAIT LOOP.  THIS IS TO ALLOW SOME DEGREE OF
;    PIPELINING TO TAKE PLACE IN THAT BY LOADING THE MDR BEFORE THE WAIT
;    LOOP INSTEAD OF AFTER IT, YOU ARE GIVING THE MEMORY AN ADDITIONAL
;    200 NSEC. TO BECOME AVAILABLE.
;*****************************************************************************

M1FULL:   RLIMM PR ALUMDR FULL
          B6 RA11 CARHO RPS ALUMAR    NCCMEMAC JMPDF OUTPUT2
WAIT2:    RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT2:  LRESWR MASHIKA JMPDF UPDATE

;*****************************************************************************
;            IF DEL < ANTI1 THEN IN FIRST HALF OF TRANSITION REGION
;                           ELSE IN SECOND HALF OF TRANSITION REGION
;*****************************************************************************

                                        ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                        ; IF RESULT < 0
                                        ;    THEN DEL < ANTI1

M1TREG:   B9 PS ALUMDR NCCNEG JMPDF M1MIN   ;ELSE RESULT >= 0
                                            ;DEL >= ANTI1
                                            ; SET UP FOR M1MIN OR M2MIN
                                            ; BY PASSING MIN TO REG. 9
```

```
;*************************************************************************
;                    FIRST HALF OF TRANSITION REGION
;              OUTPUT MEDIUM INTENSITY PIXEL IN M1 DIRECTION
;*************************************************************************

M1MED:     B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
OUTPUT3:   LRESWR DFIKD MASHIKA MED

;*************************************************************************
;              OUTPUT MINIMUM INTENSITY PIXEL IN M2 DIRECTION
;*************************************************************************

M2MIN:     B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
OUTPUT4:   LRESWR MASHIKA JMPDF UPDATE    ;    MAY BE ABLE TO USE THE MIN
                                          ;    DATA FIELD AS ORIGINALLY
                                          ;    WRITTEN SINCE WHEN THIS LOOP IS
                                          ;    USED, B9 = MIN, NOT MED

;*************************************************************************
;    AT THIS POINT, DEL > ANTI1 SO BEYOND MIDWAY POINT IN TRANSITON REGION
;              OUTPUT MINIMUM INTENSIT PIXEL IN M1 DIRECTION
;*************************************************************************

M1MIN:      B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
           LRESWR MASHIKA


;*************************************************************************
;              OUTPUT MEDIUM INTENSITY PIXEL IN M2 DIRECTION
;*************************************************************************

M2MED:     B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
           LRESWR DFIKD MASHIKA MED

;*************************************************************************
;           NOW UPDATE POSITION BY ADDING DIS1 TO BOTH COORDINATES
;                AND BY SETTING DEL = DEL + [2 * DB]
;*************************************************************************

UPDATE:    B6 RA11 CARHO RPS BD
           SQ RA0 RPS QD JMPDF DECDA      ; DEL = DEL + [2 * DB]
                                          ; GOTO DECREMENT DA (EOL COUNTER)

;*************************************************************************
;                              M2 MOVE
;       UPDATE POSITION, OUTPUT FULL INTENSITY PIXEL, UPDATE DEL
;*************************************************************************

M2MOVE:    B6 RA12 CARHO RPS ALUMAR BD NCCMEMAC JMPDF OUTPUT5 ; (Y\X)=(Y\X)+DIS2
           RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT5:    LRESWR DFIKD MASHIKA FULL  ; UPDATE DEL AS
                                       ; COMMENTED BELOW & WRITE
;*************************************************************************
;                        UPDATE DEL FOR M2 MOVE
;       DEL = DEL + [(2*DB) - (2*DA)]
;       THE VALUE ADDED TO DEL HAS ALREADY BEEN COMPUTED AND IS IN  R10
;*************************************************************************

           SQ RA10 RPS QD
```

```
; ************************************************************************
;                        DECREMENT DA (END OF LINE COUNTER)
; ************************************************************************

DECDA:    RLIMM B13 CAR1 SMR BD 2       ; DECREMENT DELTA A
                                        ; DECREMENT BY TWO FOR LORES
                                        ; HARDWARE PECULIARITY
          B13 PS BD JMPDF DRAWLN        ; SET UP FOR DELTA A CHECK
                                        ; GOTO TOP OF DRAW LOOP



END                                     ; ASSEMBLER DIRECTIVE
```

```
DEFAULT NANOP CCNOP LDNOP SB ALUZ YD CARO SSO ALUBR MDRIKD MARIKA


ORG 7000
;++
;       VECTAA2F -  LOW RESOLUTION ANTIALIASING VECT ROUTINE
;
;
;       CALLING FORMAT:
;
;               VECTAA   <VLISTADDR>
;
;       VECTOR LIST FORMAT:
;
;       WORD    UPPER 16 BITS           LOWER 16 BITS
;
;        0          Y-COORD                 X-COORD
;        1       D/M,E,SHADE          Z-COORD (IGNORED)
;       ...       ...                        ...
;
;       (CONTROL/SHADE)\Z COORD.  WORD:
;
;       BIT 31          - DRAW=1, MOVE=0
;       BIT 30          - END LIST=1, NO END LIST=0
;       BIT 29          - UNUSED
;       BIT 28          - UNUSED
;       BITS 27-24      - RESERVED
;       BITS 23-16      - VECTOR SHADE
;++
;$INCLUDE IDLSYS.EXP
;++
;       IDLSYS - IDL INTERPRETER/DISPATCHER SYSTEM CONFIGURATION EQUATES
;
;++

;++
;++


SYSFLAG=0               ;ADDRESS OF GENERAL IDL SYSTEM FLAG WORD.
ITABLE=1                ;ADDRESS OF IDL INSTRUCTION RELOC.  INDEX TABLE.
ERRWD=ITABLE+256.       ;ADDRESS OF IDL SYSTEM ERROR FLAG WORD.
INDFLAG=ERRWD+1         ;ADDRESS OF THE COPY OF THE CURRENT IDL
                        ;INSTRUCTION INDIRECT ADDRESSING FLAG WORD.
RPNT=INDFLAG+1          ;ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
SYSCOEF=RPNT+1          ;BASE ADDRESS FOR SOFTWARE TRANSFORM MATRICES
INSTR=500               ;ADDRESS OF FIRST IDL INSTRUCTION.
TSTACK=7777             ;ADDRESS OF HIGHEST SCRATCHPAD LOCATION.
BSTACK=TSTACK-500.      ;LOWEST ALLOWABLE STACK LOCATION.

;++
;       IDL RUNTIME ERROR CODE DEFINITIONS.
;++

ERR1=1                  ;IDL STACK UNDERFLOW ERROR CODE
ERR2=2                  ;IDL STACK OVERFLOW ERROR CODE.

;++
;       END OF IDL SYSTEM EQUATES
;++
;$INCLUDE IDLEQS.EXP
```

```
;++
;       IDLEQS - IDL SYSTEM ROUTINE ADDRESS EQUATES
;
;++

GPARMADR=22     ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS)
GPARMVAL=35     ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE)
GPARMSPA=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS SPECIAL)
GPARMSPV=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE SPECIAL)
IDLPUSH=54      ;IDL SYSTEM ROUTINE - PUSH Q ONTO IDL SYSTEM STACK
IDLPOP=70       ;IDL SYSTEM ROUTINE - POP Q OFF OF IDL SYSTEM STACK
IDLEXIT=3       ;IDL SYSTEM ROUTINE - EXIT AND FETCH NEXT IDL INSTRUCTION


;++
;       MISCELLANEOUS PROGRAM EQUATES.
;++

MASK8=377               ;EIGHT BIT SHADE MASK
MASK9=777               ;NINE BIT COORDINATE MASK (SEE NOTE).
MASK10=1777             ;TEN BIT MASK
MASK16=-1               ;SIXTEEN BIT MASK (HALFWORD)
YCONV=1777              ;Y COORD INVERTER VALUE
DELTA=2                 ;COORDINATE DISPLACEMENT VALUE (USED BY DIS1,DIS2)

ZERO=0                  ;CONSTANT USED FOR CLEARING THE UPPER DATA REGISTER


;++
;       MAIN PARAMETER FETCHING.
;       all IDL subroutines pass parameters though the Q register
;++

VECTAA: JSBDF GPARMVAL                          ;GET INPUT VERTEX LIST A
        SQ PS BD B1                             ;R1=VERT. PNTR.
                                                ;vertex pointer contains the
                                                ; address in scatchpad RAM of
                                                ; the next vertex word pair

        B1 RLIMM CAR1 SMR BD 1                  ;PRE-DECREMENT VERTEX POINTER

;++
;       GET NEXT VERTEX.
;++

GETN:                                   ;PASS PREVIOUS YFIN TO NEW YINIT
                                        ;PASS PREVIOUS XFIN TO NEW XINIT
                                        ;PASS PREVIOUS YFIN\XFIN
                                        ; TO NEW YINIT\XINIT


;*****************************************************************************
;           LOAD NEW YFIN\XFIN FROM SCRATCHPAD RAM TO REG.2
;*****************************************************************************

                                        ;POINT TO NEXT VERTEX
                                        ;READ NEXT VERTEX Y\X

                                        ;READ NEXT S\Z, LOAD MASK

;++
```

```
;       LOW RESOLUTION COORDINATE TRANSLATION (HARDWARE PECULIARITY).
;       NOTE:
;             THE COORDINATES ARE ACTUALLY MASKED TO 9 BITS. SINCE THE
;       COORDINATES WILL BE MULTIPLIED BY 2, THE END RESULT IS COORDINATES
;       EFFECTIVELY MASKED TO 10 BITS.
;++
                                                ;RETURN MASKED AND SHIFTED
                                                ;YFIN\XFIN TO REG. 2

;*****************************************************************************
;                    EXTRACT YFIN AND XFIN FROM REG. 2
;                AND PLACE IN REG. 4 AND REG. 5 RESPECTIVELY
;*****************************************************************************
```

```
;*****************************************************************************
;                              PREPARE FOR EXIT
;      STORE ENDPOINT AT THE ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
;*****************************************************************************

        B2 PS ALUMDR  LDUDR 202         ;MDR=CURRENT ABS. YFIN\XFIN
        RIMM PR ALUMAR  RPNT            ;MAR=202\RPNT
EOL:    IKWR JMPDF IDLEXIT              ;WRITE CURRENT POINT, THEN EXIT.

;++
;       SUBROUTINES.
;++


;++
;       VECDIS - GIVEN THE VECTOR END POINTS, VECDIS DETERMINES THE
;                CHANGES (OR DELTAS) IN THE X AND Y  COORDINATES BET-
;                WEEN POINTS OF THE VECTOR.
;
```

```
;       ON ENTRY:
;         R4=FINAL Y COORDINATE
;         R5=FINAL X COORDINATE
;         R6=YFIN\XFIN
;         R7=INITIAL X COORDINATE
;         R8=INITIAL Y COORDINATE
;
;       WORKING VARIABLES:
;         R9=DX
;         R10=DY
;         R11=DISPLACEMENT 1
;         R12=DISPLACEMENT 2
;
;       IDL SYSTEM REGISTERS - DO NOT DESTROY!
;         R14=SOFTWARE STACK POINTER
;         R15=INSTRUCTION POINTER
```

```
;*****************************************************************************
;
;       HERE BEGINS THE TOTALLY NEW CODE FOR DRAWING ANTIALIASED LINES
;
;*****************************************************************************


;*****************************************************************************
;                          REGISTER ALLOCATION
;*****************************************************************************

;                                ALLOCATED FOR:

;REGISTER       ON ENTRY                IN PROCESS          ON EXIT
;
;     0        2*DELTA(B)                 SAME             SAME
;     1        VERTEX POINTER             SAME             SAME
;     2        YFIN\XFIN                  SAME             SAME
;     3        CONTROL\SHADE\Z COORD    C\S\IMIN           SAME
;     4        YFIN                       SAME             SAME
;     5        XFIN                       SAME             SAME
;     6        YINIT\XINIT                SAME             SAME
;     7        XINIT                       DB              ANTI1
;     8        YINIT                       DB              ANTI2
;     9        DX                  FRAME BUFFER INPUT      SAME
;    10        DY                          DB              DEL2AB = 2*[DB - DA]
;    11        DIS1                       SAME             SAME
;    12        DIS2                       SAME             SAME
;    13        DELTA(A)                 (DA - N)           ZERO
;    14        SOFTWARE STACK PNTR        SAME             SAME
;    15        INSTRUCTION PNTR           SAME             SAME


;*****************************************************************************
;
;                    set shade constants (numbers in octal)
;
;    A shade value is specified as a 16-bit value which is used to access
;    a 256 place color lookup table in the RTI system.  Hardware intialization,
;    prior to running this microcode, sets the Ikonas video output channel for
;    pseudocolor red.  In this mode, only the low 8 bits of the data at
;    each pixel location are used to address the color lookup table.  This
;    color lookup table is initialized by the IDL program which calls this
;    microcode.  This initialization amounts to loading the three locations
;    in the lookup table, those addressed by the values this microcode
;    places in the frame buffer, with the appropriate intensity values.
;
;*****************************************************************************

FULL=177777     ; hex FFFF - position 255 in lookup table
MED=114631      ; hex 9999 - position 153 in color lookup
MIN=042104      ; hex 4444 - position 68 in color lookup



;*****************************************************************************
;               set up constants to be input to Ikonas 16 bit counter
;          for use in keeping track of the number of left shifts used to
```

```
;                                   implement multiplication
; ********************************************************************************

LAPCON1=177777                      ; X4 MULTIPLIER:  1'S COMPL. OF ZERO
LAPCON2=177775                      ; X16 MULTIPLIER: 1'S COMPL. OF 2
RATIO=177774                        ; X32 MULTIPLIER: 1'S COMPL. OF 3


; ********************************************************************************
;                           CLEAR UPPER DATA REGISTER
;       TO PERMIT THE USE OF IMMEDIATE DATA WITH THE DFIKD SPECIFICATION.
;                           ALSO CLEAR THE Z FIELD
; ********************************************************************************

DRAW:       LDUDR MASK16                ; CAN ALMOST CERTAINLY BE
                                        ; BE PIPELINED
            B2 RIMM RAS BD ZERO         ; CLEAR THE Z FIELD IN REG.2


            LDUDR ZERO                  ; CAN PROBABLY PIPELINE THIS STEP
                                        ; THOUGH NO SIGNIFICANT BENEFIT
                                        ; WOULD RESULT

; ********************************************************************************
;
;                           CALCULATE DEL
;
; ********************************************************************************

            RA0 B13 CAR1 RMS QD


; ********************************************************************************
;                           DETERMINE IF DEL IS POSITIVE OR NEGATIVE
; ********************************************************************************

            CCNEG JMPDF NEGDEL

; ********************************************************************************
;                   FOR DEL >= 0, SET CONSTANTS FOR DIAGONAL LINE
; ********************************************************************************

POSDEL:     RA0 CAR1 MR B7 BD           ; ANTI1 (R7) GETS -2*DELTA(B)
MINHI:      B2 RLIMM RORS BD MED        ; PUT IMIN IN Z FIELD OF REG.2
            SQ RA13 CAR1 SMR B10 BD     ; SET [(2*DB)-(2*DA)] FOR USE LATER
            JMPDF ANTI2

; ********************************************************************************
;                           SET CONSTANTS FOR AXIAL LINES
; ********************************************************************************

NEGDEL:     RA0 PR B10 RAFBD            ; D(B) = [2*D(B)]/2
            RA10 PR B8 BD               ; COPY D(B) INTO REG.8

            LDCNT RATIO                 ; SET UP MULTIPLY CONSTANT IN COUNTER
MULT:       B8 PS LAFBD NCCCNTZ JMPCNT MULT  ; MULT. RATIO * DB

            RA13 B8 CAR1 RMS            ; D(A) -- RATIO*D(B)
            RA10 PR B7 BD CCNEG JMPDF STANDRD

                                        ; PASS D(B) TO REG.7
```

```
;                                   ; IF RESULT < 0, RATIO*D(B) > D(A), STANDARD
;                                   ;          ELSE RATIO*D(B) <= D(A), SHALLOW

;*****************************************************************************
;                NOW SET CONSTANTS FOR SHALLOW AXIAL LINES
;*****************************************************************************

SHALLOW:  LDCNT LAPCON2                     ; MULTIPLY LAPCON2 * DB
SHMULT:   B7 PS LAFBD NCCCNTZ JMPCNT SHMULT

          B7 CAR1 MS BD                     ; -LAPCON2 * DB TO REG.7

          JMPDF MINLO

;*****************************************************************************
;                   SET CONSTANTS FOR STANDARD AXIAL LINES
;*****************************************************************************

STANDRD:  LDCNT LAPCON1                     ; LAPCON1 * DB
STMULT:   B7 PS LAFBD NCCCNTZ JMPCNT STMULT

          B7 CAR1 MS BD                     ; ANTI1 = -LAPCON1 * DB


;*****************************************************************************
;                SET MINIMUM TRANSITION REGION LENGTH IF DEL FALLS
;                INTO SECOND HALF OF TRANSITION REGION
;*****************************************************************************

          SQ RA7 CAR1 SMR MINLO             ; COMPARE [DEL - ANTI1]
                                            ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                            ; IF RESULT < 0 THEN DEL < ANTI1
          NCCNEG JMPRDF MINREGN             ; ELSE RESULT >= 0 AND DEL >= ANTI1

MINREGN:  RA0 CAR1 MR B7 BD                 ; SET ANTI1 = - [2*DB]

MINLO:    B2 RLIMM RORS BD MIN              ; PUT IMIN IN Z FIELD OF REG.2

;*****************************************************************************
;          SET UP THE CONSTANT [(2*DB) - (2*DA)] IN REG. 10 FOR USE LATER
;*****************************************************************************

          SQ RA13 CAR1 SMR B10 BD

;*****************************************************************************
;                              SYMMETRY CORRECTION
;*****************************************************************************

          SQ RA7 RPS QD                     ; DEL = DEL + ANTI1

;*****************************************************************************
;                                   SET ANTI2
;*****************************************************************************

ANTI2:    RA7 PR LAFBD B8                   ; ANTI2 = ANTI1 * 2


;*****************************************************************************
;                              OUTPUT THE STARTING POINT
;*****************************************************************************
```

```
          SQ RA8 CAR1 SMR PT1FULL          ; DEL - ANTI2
                                           ; ANTI2 IS ALWAYS NEGATIVE
                                           ; DEL MAYY BE POS. OR NEG. HERE
                                           ; IF DEL POS.
                                           ;    THEN RESULT > 0, DEL > ANTI2

                                           ; IF DEL NEG.
                                           ;    IF RESULT < 0, THEN DEL < ANTI2
          NCCNEG JMPRDF PT1MED             ;    ELSE RESULT >= 0, DEL >= ANTI2


PT1FULL:  RLIMM PR ALUMDR FULL             ; LOAD MDR WITH FULL INTENS. VALUE
          B6 PS ALUMAR BD CCMEMAC JMPDF .
          LRESWR MASHIKA B13 PS BD JMPDF DRAWLN ; SET UP FOR INITIAL
                                                ; DELTA A (EOL) CHECK


PT1MED:   B6 PS ALUMAR BD CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                       ; OR the pixel value at the addressed
          B9 RLIMM RORS ALUMDR MED         ; location with the MED value to be
          LRESWR MASHIKA B13 PS BD         ; SET UP FOR EOL CHECK


;*****************************************************************************
;                          DRAW THE REMAINDER OF THE LINE
;*****************************************************************************

;*****************************************************************************
;                                EXIT IF DA = 0
;*****************************************************************************

DRAWLN:   SQ PS QD CCZERO NARETN           ; SET UP FOR DEL COMPARISON

;*****************************************************************************
;                          CHECK FOR M1 MOVE ( DEL < 0 )
;*****************************************************************************

          SQ RA8 CAR1 SMR NCCNEG JMPDF M2MOVE    ; SET UP FOR COMP. DEL. & ANTI2
                                                 ; DEL .GE. 0 MEANS M2MOVE

;*****************************************************************************
;                                   M1 MOVE
;   CHECK TO SEE IF NOT YET IN TRANSITION REGION
;   IF DEL < ANTI2 THEN NOT YET IN TRANSITION REGION
;                  ELSE DEL >= ANTI2, HAVE ENTERED TRANSITION REGION
;*****************************************************************************

M1MOVE:   SQ RA7 CAR1 SMR NCCNEG JMPDF M1TREG          ; TEST [DEL-ANTI2], DEL-
                                           ; BOTH DEL AND ANTI2 NEG. HERE
                                           ; IF RESULT < 0
                                           ;    THEN DEL < ANTI2
                                           ; ELSE RESULT >= 0
                                           ;    DEL >= ANTI2

;*****************************************************************************
;
;                          WRITE OUT FULL INTENSITY PIXEL
;   LOAD MDR BEFORE ENTERING WAIT LOOP.  THIS IS TO ALLOW SOME DEGREE OF
;   PIPELINING TO TAKE PLACE IN THAT BY LOADING THE MDR BEFORE THE WAIT
```

```
;   200 NSEC. TO BECOME AVAILABLE.
;
;*****************************************************************************

M1FULL:   RLIMM PR ALUMDR FULL
          B6 RA11 CARHO RPS ALUMAR     NCCMEMAC JMPDF OUTPUT2
          RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT2:  LRESWR MASHIKA JMPDF UPDATE

;*****************************************************************************
;           IF DEL < ANTI1 THEN IN FIRST HALF OF TRANSITION REGION
;                          ELSE IN SECOND HALF OF TRANSITION REGION
;*****************************************************************************

                                       ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                       ; IF RESULT < 0
                                       ;    THEN DEL < ANTI1
M1TREG:   NCCNEG JMPDF M1MIN           ; ELSE RESULT >= 0
                                       ;    DEL >= ANTI1

;*****************************************************************************
;                      FIRST HALF OF TRANSITION REGION
;                OUTPUT MEDIUM INTENSITY PIXEL IN M1 DIRECTION
;*****************************************************************************

M1MED:    B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                   ; OR pixel value at addressed location
          B9 RLIMM RORS ALUMDR MED     ; with MED value to be output
          LRESWR MASHIKA

;*****************************************************************************
;                OUTPUT MINIMUM INTENSITY PIXEL IN M2 DIRECTION
;*****************************************************************************

M2MIN:    B2 RLIMM RAS ALUMDR MASK16   ; LOAD MDR PRIOR TO WAIT LOOP FOR
                                       ; PIPELINE EFFECT

          B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA               ;    MAY BE ABLE TO USE THE MIN
          IKBR BD B9                   ;    DATA FIELD AS ORIGINALLY
          B9 RMDR RORS ALUMDR          ;    WRITTEN SINCE WHEN THIS LOOP IS
          LRESWR MASHIKA JMPDF UPDATE  ;    USED, B9 = MIN, NOT MED ??
                                       ;    NO REASON TO CHANGE THIS HOWEVER


;*****************************************************************************
;     AT THIS POINT, DEL > ANTI1 SO BEYOND MIDWAY POINT IN TRANSITON REGION
;                OUTPUT MINIMUM INTENSIT PIXEL IN M1 DIRECTION
;*****************************************************************************

M1MIN:    B2 RLIMM RAS ALUMDR MASK16   ; LOAD MDR PRIOR TO WAIT LOOP
                                       ; FOR PIPELINE EFFECT

          B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                   ; OR pixel value at addressed location
          B9 RMDR RORS ALUMDR          ; with IMIN value to be output
          LRESWR MASHIKA                     ; there may be a wait here
```

```
                                        ; ... no delay loop need be
                                        ; explicitly specified

;*****************************************************************************
;              OUTPUT MEDIUM INTENSITY PIXEL IN M2 DIRECTION
;*****************************************************************************

M2MED:    B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
          LRESRD MASHIKA
          IKBR BD B9                    ; OR pixel value at addressed location
          B9 RLIMM RORS ALUMDR MED      ; with MED value to be output
          LRESWR MASHIKA

;*****************************************************************************
;          NOW UPDATE POSITION BY ADDING DIS1 TO BOTH COORDINATES
;               AND BY SETTING DEL = DEL + [2 * DB]
;*****************************************************************************

UPDATE:   B6 RA11 CARHO RPS BD          ; [Y\X] + M1 DISPLACEMENT (DIS1)
          SQ RAO RPS QD JMPDF DECDA     ; DEL = DEL + [2 * DB]
                                        ; GOTO DECREMENT DA (EOL COUNTER)

;*****************************************************************************
;                                M2 MOVE
;       UPDATE POSITION, OUTPUT FULL INTENSITY PIXEL, UPDATE DEL
;*****************************************************************************

M2MOVE:   B6 RA12 CARHO RPS ALUMAR BD NCCMEMAC JMPDF OUTPUT5 ; (Y\X)=(Y\X)+DIS2
          RA6 PR ALUMAR JMPDF .
OUTPUT5:  LRESWR DFIKD MASHIKA FULL

;*****************************************************************************
;                         UPDATE DEL FOR M2 MOVE
;       DEL = DEL + [(2*DB) - (2*DA)]
;       THE VALUE ADDED TO DEL HAS ALREADY BEEN COMPUTED AND IS IN  R10
;*****************************************************************************

          SQ RA10 RPS QD                ; DEL = DEL + [(2*DB) - (2*DA)]


;*****************************************************************************
;                         DECREMENT DA (END OF LINE COUNTER)
;*****************************************************************************


DECDA:    RLIMM B13 CAR1 SMR BD 2       ; DECREMENT DELTA A
                                        ; DECREMENT BY TWO FOR LORES
                                        ; HARDWARE PECULIARITY
          B13 PS BD JMPDF DRAWLN



END                                     ; ASSEMBLER DIRECTIVE
```

```
ORG 6000

;++
;       VECTEADI - LOW RESOLUTION ANTIALIASING VECT ROUTINE
;                  USING VECTAA1
;
;       CALLING FORMAT:
;
;               VECTAA   <VLISTADDR>
;
;       VECTOR LIST FORMAT:
;
;       WORD    UPPER 16 BITS           LOWER 16 BITS
;
;        0          Y-COORD                 X-COORD
;        1       D/M,E,SHADE          Z-COORD (IGNORED)
;       ...        ...                        ...
;
;       (CONTROL/SHADE)\Z COORD. WORD:
;
;       BIT 31          - DRAW=1, MOVE=0
;       BIT 30          - END LIST=1, NO END LIST=0
;       BIT 29          - UNUSED
;       BIT 28          - UNUSED
;       BITS 27-24      - RESERVED
;       BITS 23-16      - VECTOR SHADE
;++
;$INCLUDE IDLSYS.EXP
;++
;       IDLSYS - IDL INTERPRETER/DISPATCHER SYSTEM CONFIGURATION EQUATES
;
;++

;++
;++


SYSFLAG=0               ;ADDRESS OF GENERAL IDL SYSTEM FLAG WORD.
ITABLE=1                ;ADDRESS OF IDL INSTRUCTION RELOC. INDEX TABLE.
ERRWD=ITABLE+256.       ;ADDRESS OF IDL SYSTEM ERROR FLAG WORD.
INDFLAG=ERRWD+1         ;ADDRESS OF THE COPY OF THE CURRENT IDL
                        ;INSTRUCTION INDIRECT ADDRESSING FLAG WORD.
RPNT=INDFLAG+1          ;ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
SYSCOEF=RPNT+1          ;BASE ADDRESS FOR SOFTWARE TRANSFORM MATRICES
INSTR=500               ;ADDRESS OF FIRST IDL INSTRUCTION.
TSTACK=7777             ;ADDRESS OF HIGHEST SCRATCHPAD LOCATION.
BSTACK=TSTACK-500.      ;LOWEST ALLOWABLE STACK LOCATION.

;++
;       IDL RUNTIME ERROR CODE DEFINITIONS.
;++

ERR1=1                  ;IDL STACK UNDERFLOW ERROR CODE
ERR2=2                  ;IDL STACK OVERFLOW ERROR CODE.

;++
;       END OF IDL SYSTEM EQUATES
;++
;$INCLUDE IDLEQS.EXP
```

```
; ++
;       IDLEQS - IDL SYSTEM ROUTINE ADDRESS EQUATES
;
; ++

GPARMADR=22     ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS)
GPARMVAL=35     ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE)
GPARMSPA=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (ADDRESS SPECIAL)
GPARMSPV=0      ;IDL SYSTEM ROUTINE - GET PARAMETER (VALUE SPECIAL)
IDLPUSH=54      ;IDL SYSTEM ROUTINE - PUSH Q ONTO IDL SYSTEM STACK
IDLPOP=70       ;IDL SYSTEM ROUTINE - POP Q OFF OF IDL SYSTEM STACK
IDLEXIT=3       ;IDL SYSTEM ROUTINE - EXIT AND FETCH NEXT IDL INSTRUCTION


; ++
;       MISCELLANEOUS PROGRAM EQUATES.
; ++

MASK8=377               ;EIGHT BIT SHADE MASK
MASK9=777               ;NINE BIT COORDINATE MASK (SEE NOTE).
MASK10=1777             ;TEN BIT MASK
MASK16=-1               ;SIXTEEN BIT MASK (HALFWORD)
YCONV=1777              ;Y COORD INVERTER VALUE
DELTA=2                 ;COORDINATE DISPLACEMENT VALUE (USED BY DIS1,DIS2)

ZERO= 000 000           ; CONSTANT USED TO CLEAR UPPER DATA REGISTER

; ++
;       MAIN PARAMETER FETCHING.
; ++

VECTAA: JSBDF GPARMVAL                  ;GET INPUT VERTEX LIST A
        SQ PS BD B1                     ;R1=VERT. PNTR.
                                        ;vertex pointer contains the
                                        ; address in scratchpad RAM of
                                        ; the next vertex word pair

        B1 RLIMM CAR1 SMR BD 1          ;PRE-DECREMENT VERTEX POINTER

; ++
;       GET NEXT VERTEX.
; ++

GETN:                                   ;PASS PREVIOUS YFIN TO NEW YINIT
                                        ;PASS PREVIOUS XFIN TO NEW XINIT
                                        ;PASS PREVIOUS YFIN\XFIN
                                        ; TO NEW YINIT\XINIT


;*************************************************************************
;           LOAD NEW YFIN\XFIN FROM SCRATCHPAD RAM TO REG.2
;*************************************************************************

                                        ;POINT TO NEXT VERTEX
                                        ;READ NEXT VERTEX Y\X

                                        ;READ NEXT S\Z, LOAD MASK
; ++
;       MASK Y\X COORDINATES TO 10 BITS, THEN MULTIPLY BOTH BY 2 FOR
;       LOW RESOLUTION COORDINATE TRANSLATION (HARDWARE PECULIARITY).
```

```
;         THE COORDINATES ARE ACTUALLY MASKED TO 9 BITS.  SINCE THE
;       COORDINATES WILL BE MULTIPLIED BY 2, THE END RESULT IS COORDINATES
;       EFFECTIVELY MASKED TO 10 BITS.
;++
                                                ;RETURN MASKED AND SHIFTED
                                                ;YFIN\XFIN TO REG. 2

;*****************************************************************************
;                    EXTRACT YFIN AND XFIN FROM REG. 2
;                AND PLACE IN REG. 4 AND REG. 5 RESPECTIVELY
;*****************************************************************************
```



```
;*****************************************************************************
;                              PREPARE FOR EXIT
;       STORE ENDPOINT AT THE ADDRESS OF THE SYSTEM RELATIVE VERTEX WORD
;*****************************************************************************

        B2 PS ALUMDR  LDUDR 202         ;MDR=CURRENT ABS.  YFIN\XFIN
        RIMM PR ALUMAR  RPNT            ;MAR=202\RPNT
EOL:    IKWR JMPDF IDLEXIT              ;WRITE CURRENT POINT,  THEN EXIT.

;++
;       SUBROUTINES.
;++


;++
;       VECDIS - GIVEN THE VECTOR END POINTS, VECDIS DETERMINES THE
;                CHANGES (OR DELTAS) IN THE X AND Y  COORDINATES BET-
;                WEEN POINTS OF THE VECTOR.
;
;       ON ENTRY:
;         R4=FINAL Y COORDINATE
```

```
;         R6=YFIN\XFIN
;         R7=INITIAL X COORDINATE
;         R8=INITIAL Y COORDINATE
;
;       WORKING VARIABLES:
;         R9=DX
;         R10=DY
;         R11=DISPLACEMENT 1
;         R12=DISPLACEMENT 2
;
;       IDL SYSTEM REGISTERS - DO NOT DESTROY!
;         R14=SOFTWARE STACK POINTER
;         R15=INSTRUCTION POINTER
```

```
;
;       HERE BEGINS THE TOTALLY NEW CODE FOR DRAWING ANTIALIASED LINES
;
;*****************************************************************************

;*****************************************************************************
;                          REGISTER ALLOCATION
;*****************************************************************************

;                                ALLOCATED FOR:

;REGISTER        ON ENTRY                    IN PROCESS          ON EXIT
;
;     0        2*DELTA(B)                     SAME               SAME
;     1        VERTEX POINTER                 SAME               SAME
;     2        YFIN\XFIN                      SAME               SAME
;     3        CONTROL\SHADE\Z COORD          SAME               SAME
;     4        YFIN                           SAME               SAME
;     5        XFIN                           SAME               SAME
;     6        YINIT\XINIT                    SAME               SAME
;     7        XINIT                          DB                 ANTI1
;     8        YINIT                          DB                 ANTI2
;     9        DX                             SAME               IMIN
;    10        DY                             DB                 DEL2AB = 2*[DB - DA]
;    11        DIS1                           SAME               SAME
;    12        DIS2                           SAME               SAME
;    13        DELTA(A)                       (DA - N)           ZERO
;    14        SOFTWARE STACK PNTR            SAME               SAME
;    15        INSTRUCTION PNTR               SAME               SAME

;*****************************************************************************
;
;                    set shade constants (numbers in octal)
;
;    A shade value is specified as a 16-bit value which is used to access
;    a 256 place color lookup table in the RTI system.  Hardware initialization,
;    prior to running this microcode, sets the Ikonas video output channel for
;    pseudocolor red.  In this mode, only the low 8 bits of the data at
;    each pixel location are used to address the color lookup table.  This
;    color lookup table is initialized by the IDL program which calls this
;    microcode.  This initialization amounts to loading the three locations
;    in the lookup table, those addressed by the values this microcode
;    places in the frame buffer, with the appropriate intensity values.
;
;*****************************************************************************


FULL=000335       ; hex 00DD - position 13 in LU table, full White
MED=114631        ; hex 9999 - position 10 in LU table, med white (grey)

MIN=042104        ; hex 4444 - position 5 in LU table, min White (light grey)


;*****************************************************************************
;                set up constants to be input to Ikonas 16 bit counter
;             for use in keeping track of the number of left shifts used to
;                                  implement multiplication
;*****************************************************************************
```

```
LAPCON1=177777                  ; X4 MULTIPLIER:   1'S COMPL. OF ZERO
LAPCON2=177775                  ; X16 MULTIPLIER:  1'S COMPL. OF 2
RATIO=177774                    ; X32 MULTIPLIER:  1'S COMPL. OF 3

;*************************************************************************
;                       CLEAR UPPER DATA REGISTER
;*************************************************************************


;*************************************************************************
;
;                       CALCULATE DEL
;
;*************************************************************************


DRAW:     RA0 B13 CAR1 RMS QD LDUDR ZERO

;*************************************************************************
;                       DETERMINE IF DEL IS POSITIVE OR NEGATIVE
;*************************************************************************

          CCNEG JMPDF NEGDEL

;*************************************************************************
;               FOR DEL >= 0, SET CONSTANTS FOR DIAGONAL LINE
;*************************************************************************

POSDEL:   RA0 CAR1 MR B7 BD             ; ANTI1 (R7) GETS -2*DELTA(B)
MINHI:    RLIMM PR B9 BD MED            ; PASS MED. INTENSITY VALUE TO MIN REG.
          SQ RA13 CAR1 SMR B10 BD JMPDF ANTI2    ; [(2*DB)-(2*DA)] FOR USE LATER

;*************************************************************************
;                       SET CONSTANTS FOR AXIAL LINES
;*************************************************************************

NEGDEL:   RA0 PR B10 RAFBD              ; D(B) = [2*D(B)]/2
          RA10 PR B8 BD                 ; COPY D(B) INTO REG.8

          LDCNT RATIO                   ; SET UP MULTIPLY CONSTANT IN COUNTER
MULT:     B8 PS LAFBD NCCCNTZ JMPCNT MULT  ; MULT. RATIO * DB

          RA13 B8 CAR1 RMS              ; D(A) - RATIO*D(B)

          RA10 PR B7 BD CCNEG JMPDF STANDRD
                                        ; pass D(B) to reg.7
                                        ; IF RESULT < 0, RATIO*D(B) > D(A)
                                        ;       THEN STANDARD
                                        ; ELSE RES. >= 0, RATIO*D(B) <= D(A)
          ;

;*************************************************************************
;               NOW SET CONSTANTS FOR SHALLOW AXIAL LINES
;*************************************************************************

SHALLOW:  LDCNT LAPCON2                 ; MULTIPLY LAPCON2 * DB
SHMULT:   B7 PS LAFBD NCCCNTZ JMPCNT SHMULT

          B7 CAR1 MS BD JMPDF MINLO                  ; -LAPCON2 * DB TO REG.7
```

```
;*************************************************************************
;                  SET CONSTANTS FOR STANDARD AXIAL LINES
;*************************************************************************

STANDRD:  LDCNT LAPCON1                    ; LAPCON1 * DB
STMULT:   B7 PS LAFBD NCCCNTZ JMPCNT STMULT

          B7 CAR1 MS BD                    ; ANTI1 = -LAPCON1 * DB


;*************************************************************************
;              SET MINIMUM TRANSITION REGION LENGTH IF DEL FALLS
;              INTO SECOND HALF OF TRANSITION REGION
;*************************************************************************

          SQ RA7 CAR1 SMR MINLO            ; COMPARE [DEL - ANTI1]
                                           ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                           ; IF RESULT < 0 THEN DEL < ANTI1
          NCCNEG JMPRDF MINREGN            ; ELSE RESULT >= 0 AND DEL >= ANTI1

MINREGN:  RA0 CAR1 MR B7 BD                ; SET ANTI1 = - [2*DB]

MINLO:    RLIMM PR B9 BD MIN               ; PASS LOW INTENSITY VALUE TO MIN REG.

;*************************************************************************
;          SET UP THE CONSTANT [(2*DB) - (2*DA)] IN REG. 10 FOR USE LATER
;*************************************************************************

          SQ RA13 CAR1 SMR B10 BD

;*************************************************************************
;                              SYMMETRY CORRECTION
;*************************************************************************

          SQ RA7 RPS QD                    ; DEL = DEL + ANTI1

;*************************************************************************
;                                 SET ANTI2
;*************************************************************************

ANTI2:    RA7 PR LAFBD B8                  ; ANTI2 = ANTI1 * 2


;*************************************************************************
;                           OUTPUT THE STARTING POINT
;*************************************************************************

          SQ RA8 CAR1 SMR PT1FULL          ; DEL - ANTI2
                                           ; ANTI2 IS ALWAYS NEG.
                                           ; DEL MAY BE POS. OR NEG. HERE
                                           ; IF DEL POS.
                                           ;    THEN RESULT > 0, DEL > ANTI2

                                           ; IF DEL NEG. THEN
                                           ;    IF RESULT < 0, THEN DEL < ANTI2
          NCCNEG JMPRDF PT1MED             ;    ELSE RESULT >= 0 , DEL >= ANTI2

PT1FULL:  RLIMM PR ALUMDR FULL             ; LOAD MDR WITH FULL INTENS. VALUE
          B6 PS ALUMAR BD CCMEMAC JMPDF .
```

```
OUTPUT1:   LRESWR MASHIKA B13 PS BD JMPDF DRAWLN  ; SET UP FOR INITIAL
                                                  ; DELTA A (EOL) CHECK

PT1MED:    RLIMM PR ALUMDR MED            ; LOAD MDR W. MEDIUM INTENS. VALUE
           B6 PS ALUMAR BD CCMEMAC JMPDF .
           LRESWR MASHIKA B13 PS BD       ; SET UP FOR EOL CHECK

;*****************************************************************************
;                          DRAW THE REMAINDER OF THE LINE
;*****************************************************************************

;*****************************************************************************
;                                EXIT IF DA = 0
;*****************************************************************************

                                          ; EXIT IF D(A) IS ZERO

DRAWLN:    SQ PS QD CCZERO NARETN         ; SET UP FOR DEL COMPARISON
;*****************************************************************************
;                          CHECK FOR M1 MOVE ( DEL < 0 )
;*****************************************************************************

           SQ RA8 CAR1 SMR NCCNEG JMPDF M2MOVE   ; SET UP FOR COMP. DEL & ANTI2
                                                 ; DEL .GE. 0 MEANS M2MOVE

;*****************************************************************************
;                                  M1 MOVE
;                    CHECK TO SEE IF NOT YET IN TRANSITION REGION
;          IF DEL < ANTI2 THEN NOT YET IN TRANSITION REGION
;                           ELSE DEL >= ANTI2, HAVE ENTERED TRANSITION REGION
;*****************************************************************************

                                          ; BOTH DEL AND ANTI2 NEG. HERE
M1MOVE:    SQ RA7 CAR1 SMR NCCNEG JMPDF M1TREG  ; TEST [DEL-ANTI2], DEL-ANTI1
                                          ; IF RESULT < 0
                                          ;     THEN DEL < ANTI2
                                          ; ELSE RESULT >= 0
                                          ;     DEL >= ANTI2

;*****************************************************************************
;                          WRITE OUT FULL INTENSITY PIXEL
;    LOAD MDR BEFORE ENTERING WAIT LOOP.  THIS IS TO ALLOW SOME DEGREE OF
;    PIPELINING TO TAKE PLACE IN THAT BY LOADING THE MDR BEFORE THE WAIT
;    LOOP INSTEAD OF AFTER IT, YOU ARE GIVING THE MEMORY AN ADDITIONAL
;    200 NSEC. TO BECOME AVAILABLE.
;*****************************************************************************

M1FULL:    RLIMM PR ALUMDR FULL
           B6 RA11 CARHO RPS ALUMAR    NCCMEMAC JMPDF OUTPUT2
WAIT2:     RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT2:   LRESWR MASHIKA JMPDF UPDATE   ; CAN PROB. PIPELINE

;*****************************************************************************
;           IF DEL < ANTI1 THEN IN FIRST HALF OF TRANSITION REGION
;                          ELSE IN SECOND HALF OF TRANSITION REGION
;*****************************************************************************

                                          ; BOTH DEL AND ANTI1 ARE NEG. HERE
                                          ; IF RESULT < 0
```

```
                                                ; THEN DEL < ANTI1

M1TREG:    B9 PS ALUMDR NCCNEG JMPDF M1MIN     ;ELSE RESULT >= 0
                                               ;DEL >= ANTI1
                                               ; SET UP FOR M1MIN OR M2MIN
                                               ; BY PASSING MIN TO REG. 9

;*************************************************************************
;                    FIRST HALF OF TRANSITION REGION
;               OUTPUT MEDIUM INTENSITY PIXEL IN M1 DIRECTION
;*************************************************************************

M1MED:     B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
OUTPUT3:   LRESWR DFIKD MASHIKA MED        ; NOT SURE BUT PROB. OK

;*************************************************************************
;               OUTPUT MINIMUM INTENSITY PIXEL IN M2 DIRECTION
;*************************************************************************

M2MIN:     B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
OUTPUT4:   LRESWR MASHIKA JMPDF UPDATE     ; ?? MAY BE ABLE TO USE THE MIN
                                           ;    DATA FIELD AS ORIGINALLY
                                           ;    WRITTEN SINCE WHEN THIS LOOP IS
                                           ;    USED, B9 = MIN, NOT MED

;*************************************************************************
;    AT THIS POINT, DEL > ANTI1 SO BEYOND MIDWAY POINT IN TRANSITON REGION
;               OUTPUT MINIMUM INTENSIT PIXEL IN M1 DIRECTION
;*************************************************************************

M1MIN:      B6 RA11 CARHO RPS ALUMAR CCMEMAC JMPDF .
           LRESWR MASHIKA


;*************************************************************************
;               OUTPUT MEDIUM INTENSITY PIXEL IN M2 DIRECTION
;*************************************************************************

M2MED:     B6 RA12 CARHO RPS ALUMAR CCMEMAC JMPDF .
           LRESWR DFIKD MASHIKA MED

;*************************************************************************
;          NOW UPDATE POSITION BY ADDING DIS1 TO BOTH COORDINATES
;               AND BY SETTING DEL = DEL + [2 * DB]
;*************************************************************************

UPDATE:    B6 RA11 CARHO RPS BD
           SQ RA0 RPS QD JMPDF DECDA       ; DEL = DEL + [2 * DB]
                                           ; GOTO DECREMENT DA (EOL COUNTER)

;*************************************************************************
;                              M2 MOVE
;       UPDATE POSITION, OUTPUT FULL INTENSITY PIXEL, UPDATE DEL
;*************************************************************************

M2MOVE:    B6 RA12 CARHO RPS ALUMAR BD NCCMEMAC JMPDF OUTPUT5 ; (Y\X)=(Y\X)+DIS2
           RA6 PR ALUMAR CCMEMAC JMPDF .

OUTPUT5:    LRESWR DFIKD MASHIKA FULL  ; UPDATE DEL AS
                                                ; COMMENTED BELOW & WRITE
;*************************************************************************
```

```
;       THE VALUE ADDED TO DEL HAS ALREADY BEEN COMPUTED AND IS IN  R10
; ***************************************************************************

          SQ RA10 RPS QD

; ***************************************************************************
;                          DECREMENT DA (END OF LINE COUNTER)
; ***************************************************************************

DECDA:    RLIMM B13 CAR1 SMR BD 2       ; DECREMENT DELTA A
                                        ; DECREMENT BY TWO FOR LORES
                                        ; HARDWARE PECULIARITY
          B13 PS BD JMPDF DRAWLN        ; SET UP FOR DELTA A CHECK
                                        ; GOTO TOP OF DRAW LOOP

; ***************************************************************************
;                              RETURN TO CALLING PROGRAM
; ***************************************************************************

EXIT:     NARETN
                                        ; can be pipelined w. previously
                                        ; EXECUTED step.


END                                     ; ASSEMBLER DIRECTIVE
```

```
;                          AA1GRID.IDL
;
      .RADIX 8
      .LIST 1
VECTAA1=11
VECTAA2=12
BLACK=0\0
MINR=0\0524
MEDR=0\1250
FULLR=0\1774
;
BLUE=37700\0
MEDGREEN=12\170000
;
MIN_RED_ADDR=77
MED_RED_ADDR=177
FULL_RED_ADDR=377
CMAP=203\0
;
;
      MOVE #BLACK,#CMAP
      MOVE #BLUE,#CMAP+^X44
      MOVE #MEDGREEN,#CMAP+^X99
      MOVE #FULLR,#CMAP+^XFF
;
;
START:  USER #VECTAA1,#VECTS
        JMP START
;
;
VECTS:  .VECT3D
;
;   FIRST QUADRANT
;
         250,250,0,BLACK,M
         250,500,0,,D
         250,250,,,M
        255,500,0,,D
         250,250,,,M
        260,500,0,,D
        250,250,0,,M
        370,500,0,,D
        250,250,0,,M
        450,500,0,,D
        250,250,0,,M
        500,500,0,,D
        250,250,0,,M
        500,450,0,,D
        250,250,0,,M
        500,370,0,,D
        250,250,0,,M
        500,260,0,,D
        250,250,0,,M
        500,255,0,,D
        250,250,0,,M
        500,250,0,,D
;
;   FOURTH QUADRANT
;
        250,250,0,,M
        500,245,0,,D
```

```
        250,250,0,,M
        500,240,0,,D
        250,250,0,,M
        500,130,0,,D
        250,250,0,,M
        500,50,,,D
        250,250,0,,M
        500,0,,,D
        250,250,,,M
        450,0,,,D
        250,250,,,M
        370,0,,,D
        250,250,,,M
        260,0,,,D
        250,250,,,M
        255,0,,,D
        250,250,,,M
        250,0,,,D
;
;   THIRD QUADRANT
;
         250,250,,,M
        245,0,,,D
         250,250,,,M
         240,0,,,D
         250,250,,,M
        130,0,,,D
        250,250,,,M
        50,0,,,D
         250,250,,,M
         0,0,,,D
        250,250,,,M
        0,50,,,D
        250,250,,,M
         0,130,,,D
        250,250,,,M
         0,240,,,D
         250,250,,,M
        0,245,,,D
        250,250,,,M
        0,250,,,D
;
;   SECOND QUADRANT
;
        250,250,,,M
        0,255,,,D
        250,250,,,M
        0,260,,,D
        250,250,,,M
        0,370,,,D
        250,250,,,M
        0,450,,,D
        250,250,,,M
        0,500,,,D
        250,250,,,M
        50,500,,,D
        250,250,,,M
        130,500,,,D
        250,250,,,M
        240,500,,,D
        250,250,,,M
```

```
245, 500, , , D, END
;
;
. ENDD
. END
```

```
;                                   AA1GRID2.IDL
;
        .RADIX 8
        .LIST 1
VECTAA1=11
VECTAA2=12
BLACK=0\0
MINR=0\1250
MEDR=0\1522
FULLR=0\1774
;
BLUE=37700\0
MEDGREEN=12\170000
;
MIN_RED_ADDR=77
MED_RED_ADDR=177
FULL_RED_ADDR=377
CMAP=203\0
;
;
        MOVE #BLACK,#CMAP
        MOVE #MINR,#CMAP+^X44
        MOVE #MEDR,#CMAP+^X99
        MOVE #FULLR,#CMAP+^XFF
;
;
START:  USER #VECTAA1,#VECTS
        JMP START
;
;
VECTS:  .VECT3D
;
;   FIRST QUADRANT
;
          250,250,0,BLACK,M
          250,500,0,,D
          250,250,,,M
         255,500,0,,D
          250,250,,,M
         260,500,0,,D
         250,250,0,,M
         370,500,0,,D
         250,250,0,,M
         450,500,0,,D
         250,250,0,,M
         500,500,0,,D
         250,250,0,,M
         500,450,0,,D
         250,250,0,,M
         500,370,0,,D
         250,250,0,,M
         500,260,0,,D
         250,250,0,,M
         500,255,0,,D
         250,250,0,,M
         500,250,0,,D
;
;   FOURTH QUADRANT
;
         250,250,0,,M
         500,245,0,,D
```

```
250, 250, 0, , M
500, 240, 0, , D
250, 250, 0, , M
500, 130, 0, , D
250, 250, 0, , M
500, 50, , , D
250, 250, 0, , M
500, 0, , , D
250, 250, , , M
450, 0, , , D
250, 250, , , M
370, 0, , , D
250, 250, , , M
260, 0, , , D
250, 250, , , M
255, 0, , , D
250, 250, , , M
250, 0, , , D
;
;  THIRD QUADRANT
;
 250, 250, , , M
245, 0, , , D
 250, 250, , , M
 240, 0, , , D
 250, 250, , , M
130, 0, , , D
250, 250, , , M
50, 0, , , D
 250, 250, , , M
 0, 0, , , D
250, 250, , , M
0, 50, , , D
250, 250, , , M
 0, 130, , , D
250, 250, , , M
 0, 240, , , D
 250, 250, , , M
0, 245, , , D
250, 250, , , M
0, 250, , , D
;
;  SECOND QUADRANT
;
250, 250, , , M
0, 255, , , D
250, 250, , , M
0, 260, , , D
250, 250, , , M
0, 370, , , D
250, 250, , , M
0, 450, , , D
250, 250, , , M
0, 500, , , D
250, 250, , , M
50, 500, , , D
250, 250, , , M
130, 500, , , D
250, 250, , , M
240, 500, , , D
250, 250, , , M
```

```
245, 500, , , D, END
;
;
. ENDD
. END
```

```
;                                   AA2GRID. IDL
;
      .RADIX 8
      .LIST 1
VECTAA1=11
VECTAA2=12
BLACK=0\0
MINR=0\0524
MEDR=0\1250
FULLR=0\1774
;
BLUE=37700\0
MEDGREEN=12\170000
;
MIN_RED_ADDR=77
MED_RED_ADDR=177
FULL_RED_ADDR=377
CMAP=203\0
;
;
      MOVE #BLACK, #CMAP
      MOVE #BLUE, #CMAP+^X44
      MOVE #MEDGREEN, #CMAP+^X99
      MOVE #FULLR, #CMAP+^XFF
;
;
START:  USER #VECTAA2, #VECTS
        JMP START
;
;
VECTS:  .VECT3D
;
;   FIRST QUADRANT
;
         250, 250, 0, BLACK, M
         250, 500, 0, , D
         250, 250, , , M
        255, 500, 0, , D
         250, 250, , , M
        260, 500, 0, , D
        250, 250, 0, , M
        370, 500, 0, , D
        250, 250, 0, , M
        450, 500, 0, , D
        250, 250, 0, , M
        500, 500, 0, , D
        250, 250, 0, , M
        500, 450, 0, , D
        250, 250, 0, , M
        500, 370, 0, , D
        250, 250, 0, , M
        500, 260, 0, , D
        250, 250, 0, , M
        500, 255, 0, , D
        250, 250, 0, , M
        500, 250, 0, , D
;
;   FOURTH QUADRANT
;
        250, 250, 0, , M
        500, 245, 0, , D
```

```
        250,250,0,,M
        500,240,0,,D
        250,250,0,,M
        500,130,0,,D
        250,250,0,,M
        500,50,,,D
        250,250,0,,M
        500,0,,,D
        250,250,,,M
        450,0,,,D
        250,250,,,M
        370,0,,,D
        250,250,,,M
        260,0,,,D
        250,250,,,M
        255,0,,,D
        250,250,,,M
        250,0,,,D
;
;   THIRD QUADRANT
;
         250,250,,,M
        245,0,,,D
         250,250,,,M
         240,0,,,D
         250,250,,,M
        130,0,,,D
        250,250,,,M
        50,0,,,D
         250,250,,,M
         0,0,,,D
        250,250,,,M
        0,50,,,D
        250,250,,,M
         0,130,,,D
        250,250,,,M
         0,240,,,D
         250,250,,,M
        0,245,,,D
        250,250,,,M
        0,250,,,D
;
;   SECOND QUADRANT
;
        250,250,,,M
        0,255,,,D
        250,250,,,M
        0,260,,,D
        250,250,,,M
        0,370,,,D
        250,250,,,M
        0,450,,,D
        250,250,,,M
        0,500,,,D
        250,250,,,M
        50,500,,,D
        250,250,,,M
        130,500,,,D
        250,250,,,M
        240,500,,,D
        250,250,,,M
```

```
245, 500, , , D, END
;
;
. ENDD
. END
```

```
;
;
      .RADIX 8
      .LIST 1
VECTAA1=11
VECTAA2=12
BLACK=0\0
MINR=0\1250
MEDR=0\1522
FULLR=0\1774
;
BLUE=37700\0
MEDGREEN=12\170000
;
MIN_RED_ADDR=77
MED_RED_ADDR=177
FULL_RED_ADDR=377
CMAP=203\0
;
;
      MOVE #BLACK,#CMAP
      MOVE #MINR,#CMAP+^X44
      MOVE #MEDR,#CMAP+^X99
      MOVE #FULLR,#CMAP+^XFF
;
;
START:  USER #VECTAA2,#VECTS
        JMP START
;
;
VECTS:  .VECT3D
;
;   FIRST QUADRANT
;
         250,250,0,BLACK,M
         250,500,0,,D
         250,250,,,M
        255,500,0,,D
         250,250,,,M
        260,500,0,,D
        250,250,0,,M
        370,500,0,,D
        250,250,0,,M
        450,500,0,,D
        250,250,0,,M
        500,500,0,,D
        250,250,0,,M
        500,450,0,,D
        250,250,0,,M
        500,370,0,,D
        250,250,0,,M
        500,260,0,,D
        250,250,0,,M
        500,255,0,,D
        250,250,0,,M
        500,250,0,,D
;
;   FOURTH QUADRANT
;
        250,250,0,,M
        500,245,0,,D
```

```
        250, 250, 0, , M
        500, 240, 0, , D
        250, 250, 0, , M
        500, 130, 0, , D
        250, 250, 0, , M
        500, 50, , , D
        250, 250, 0, , M
        500, 0, , , D
        250, 250, , , M
        450, 0, , , D
        250, 250, , , M
        370, 0, , , D
        250, 250, , , M
        260, 0, , , D
        250, 250, , , M
        255, 0, , , D
        250, 250, , , M
        250, 0, , , D
;
;   THIRD QUADRANT
;
         250, 250, , , M
        245, 0, , , D
         250, 250, , , M
         240, 0, , , D
         250, 250, , , M
        130, 0, , , D
        250, 250, , , M
        50, 0, , , D
         250, 250, , , M
         0, 0, , , D
        250, 250, , , M
        0, 50, , , D
        250, 250, , , M
         0, 130, , , D
        250, 250, , , M
         0, 240, , , D
         250, 250, , , M
        0, 245, , , D
        250, 250, , , M
        0, 250, , , D
;
;   SECOND QUADRANT
;
        250, 250, , , M
        0, 255, , , D
        250, 250, , , M
        0, 260, , , D
        250, 250, , , M
        0, 370, , , D
        250, 250, , , M
        0, 450, , , D
        250, 250, , , M
        0, 500, , , D
        250, 250, , , M
        50, 500, , , D
        250, 250, , , M
        130, 500, , , D
        250, 250, , , M
        240, 500, , , D
        250, 250, , , M
```

```
245,500,,,D,END
;
;
.ENDD
.END
```

```
;                                LR2GRID.IDL
;
      .RADIX 8
      .LIST 1
VECTAA1=11
VECTAA2=12
BLACK=0\0
MINR=0\0524
MEDR=0\1250
FULLR=0\1774
;
BLUE=37700\0
MEDGREEN=12\170000
;
MIN_RED_ADDR=77
MED_RED_ADDR=177
FULL_RED_ADDR=377
;
FULL_ADDR=376
;
CMAP=203\0
;
;
      MOVE #BLACK,#CMAP
      MOVE #BLUE,#CMAP+^X44
      MOVE #MEDGREEN,#CMAP+^X99
      MOVE #FULLR,#CMAP+^XFF
;
            MOVE #FULLR,#CMAP+^XFE
;
;
START:  VECT #VECTS
        JMP START
;
;
VECTS:  .VECT3D
;
;
;       VERTICAL FULL INTENSITY LINES
;
         25,500,0,FULL_ADDR,M
         25,0,,,D
         75,500,,,M
         75,0,,,D
         125,500,,,M
         125,0,,,D
         175,500,,,M
         175,0,,,D
         225,500,,,M
         225,0,,,D
         275,500,,,M
         275,0,,,D
         325,500,,,M
         325,0,,,D
         375,500,,,M
         375,0,,,D
         425,500,,,M
         425,0,,,D
         475,500,,,M
         475,0,,,D,END
```

```
245, 500, , , D, END
;
;
. ENDD
. END
```

```
        250,250,0,,M
        500,240,0,,D
        250,250,0,,M
        500,130,0,,D
        250,250,0,,M
        500,50,,,D
        250,250,0,,M
        500,0,,,D
        250,250,,,M
        450,0,,,D
        250,250,,,M
        370,0,,,D
        250,250,,,M
        260,0,,,D
        250,250,,,M
        255,0,,,D
        250,250,,,M
        250,0,,,D
;
;   THIRD QUADRANT
;
         250,250,,,M
        245,0,,,D
         250,250,,,M
         240,0,,,D
         250,250,,,M
        130,0,,,D
        250,250,,,M
        50,0,,,D
         250,250,,,M
         0,0,,,D
        250,250,,,M
        0,50,,,D
        250,250,,,M
         0,130,,,D
        250,250,,,M
         0,240,,,D
         250,250,,,M
        0,245,,,D
        250,250,,,M
        0,250,,,D
;
;   SECOND QUADRANT
;
        250,250,,,M
        0,255,,,D
        250,250,,,M
        0,260,,,D
        250,250,,,M
        0,370,,,D
        250,250,,,M
        0,450,,,D
        250,250,,,M
        0,500,,,D
        250,250,,,M
        50,500,,,D
        250,250,,,M
        130,500,,,D
        250,250,,,M
        240,500,,,D
        250,250,,,M
```

```
;                              LRGRID.IDL
;
        .RADIX 8
        .LIST 1
VECTAA1=11
VECTAA2=12
BLACK=0\0
MINR=0\0524
MEDR=0\1250
FULLR=0\1774
;
BLUE=37700\0
MEDGREEN=12\170000
;
MIN_RED_ADDR=77
MED_RED_ADDR=177
FULL_RED_ADDR=377
CMAP=203\0
;
;
        MOVE #BLACK,#CMAP
        MOVE #BLUE,#CMAP+^X44
        MOVE #MEDGREEN,#CMAP+^X99
        MOVE #FULLR,#CMAP+^XFF
;
;
START:  VECT #VECTS
        JMP START
;
;
VECTS:  .VECT3D
;
;   FIRST QUADRANT
;
          250,250,0,FULL_RED_ADDR,M
          250,500,0,,D
          250,250,,,M
         255,500,0,,D
          250,250,,,M
         260,500,0,,D
         250,250,0,,M
         370,500,0,,D
         250,250,0,,M
         450,500,0,,D
         250,250,0,,M
         500,500,0,,D
         250,250,0,,M
         500,450,0,,D
         250,250,0,,M
         500,370,0,,D
         250,250,0,,M
         500,240,0,,D
         250,250,0,,M
         500,255,0,,D
         250,250,0,,M
         500,250,0,,D
;
;   FOURTH QUADRANT
;
         250,250,0,,M
         500,245,0,,D
```

```
.ENDD
.END
```